CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 17/60, 17/30, 17/20, H04L 12/58

A1

(11) International Publication Number: WO 97/46962

(43) International Publication Date: 11 December 1997 (11.12.97)

(21) International Application Number: PCT/US97/09161

(22) International Filing Date: 30 May 1997 (30.05.97)

(30) Priority Data:

60/019,264	7 June 1996 (07.06.96)	US
08/866,196	30 May 1997 (30.05.97)	US

(71) Applicant: AT & T CORP. [US/US]; 32 Avenue of the Americas, New York, NY 10013-2412 (US).

(72) Inventors: KNOWLES, Kimberly, A.; 12 Pruner Farm Road, Lebanon, NJ 08833 (US). LEWIS, David, Dolan; Apartment 6, 532 Spruce Street, Philadelphia, PA 19106 (US).

(74) Agent: DWORETSKY, Samuel, H.; AT & T Corp., P.O. Box 4110, Middletown, NJ 07748-4110 (US).

(81) Designated States: CA, JP, MX, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

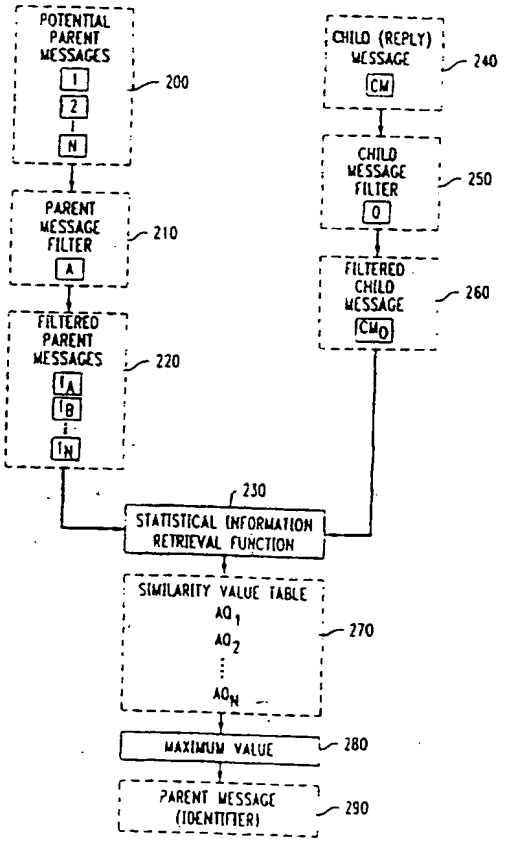
Published

With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: FINDING AN E-MAIL MESSAGE TO WHICH ANOTHER E-MAIL MESSAGE IS A RESPONSE

(57) Abstract

Current tools for processing e-mail and other messages do not adequately recognize and manipulate threads, i.e., conversations among two or more people carried out by exchange of messages. The present invention utilizes the textual context and characteristics of messages in order to provide a more reliable and effective way to construct message threads. In accordance with the present invention, statistical information retrieval techniques are used in conjuction with textual material obtained by "filtering" of messages to achieve a significant level of accuracy at identifying when one message is a reply to another.



* (Referred to in PCT Gazette No. 10/1998, Section II)

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

FINDING AN E-MAIL MESSAGE TO WHICH ANOTHER E-MAIL MESSAGE IS A RESPONSE

Cross-references to related applications

This application claims the benefit of U.S. Provisional Application No. 60/019264, filed June 7, 1996, entitled "Finding an E-mail Message to Which Another E-mail Message Is a Response."

Appendix

An attached appendix (pages 25 - 80) has been provided which lists the source code of the programs developed to carry out the experiments described below in connection with the present invention.

Copyright Notice

Technical Field

This invention relates to electronic messaging and, more particularly, to a way of recognizing and manipulating

threads contained in electronic messages.

Background of the Invention

The volume of electronic messages, such as electronic mail ("e-mail"), is huge and growing. Many users receive more messages than they can handle, which has sparked interest in better message handling software. Almost all e-mail readers now support separating messages into folders, and often allow rules to be defined to do this automatically. Tools for prioritizing and searching messages are also becoming available.

A problem with most such approaches is that they process each message individually. Many messages are parts of larger conversations, or threads. A thread is a conversation among two or more participants carried out by exchange of messages. Treating messages outside of this context may lead to undesirable results. For instance, a system that sorts messages into folders based on their content is unlikely to be 100% accurate. The effectiveness of content-based text categorization systems varies considerably among categories, and accuracies over 95% are rarely reported. This means that threads having as few as 20 component messages will almost always be broken up and distributed into multiple folders by such a system, making it difficult for a reader to follow the conversational structure.

On the other hand, a mail reading interface that understood threads could save users considerable effort. For instance, some programs for reading Usenet news allow users to delete an entire thread at once, greatly reducing the number of messages the user must inspect.

Messaging systems that are explicitly oriented to group discussion, e.g., the Usenet network and other bulletin board systems, provide the most support for threading. For instance, the reply command in most Usenet news posting programs inserts into a reply or *child* message two forms of information about the relationship between it

and its *parent* message (the message it is a reply to). First, the chain of unique message identifiers in the References: field of the parent is copied into the References: field of the child, with the unique identifier of the parent added. Second, the Subject: line of the parent is copied into the Subject: line of the child, typically prefixed by *Re:*. Usenet news readers providing a threaded display use the structural links from the References: field, while others organize a threaded display around Subject: lines which are identical or have identical prefixes.

Conversations, including group discussions, can also be carried out over electronic mail systems. The ability to send to and reply to groups of people, as well as the use of centralized mail "reflectors" and mailing list management software, can informally support multiple large scale discussions. As with bulletin board systems, replying to an e-mail message often inserts structural information into the reply. For Internet-based mail systems, the reply command may copy the Message-Id: field or other identifying information from the parent, into the In-Reply-To: field of the child. As in Usenet messages, the Subject: line is typically copied to the Subject: field, preceded by *Re:*.

Some mail clients provide threaded displays, though this is less common than for bulletin board systems. For instance, the VM mail reader (available at ftp.uu.net in networking/mail/vm directory) allows grouping of messages by one of several criteria, including having the same subject line text, the same author, or the same recipient. The mail archiving program *hypermail* (see http://www.eit.com/software/hypermail.html) marks up

archives of e-mail with a variety of links, including threading information. It attempts first to find a message id in the In-Reply-To: field and match it to a known message. Failing that it looks for a matching date string in the In-Reply-To: field, and finally tries for a match on the Subject: line, after removing one *Re:* tag.

However, the error rate of each of the above approaches is considerable. While the References: field is in theory required for replies to Usenet messages, threading is hampered by clients that delete portions of the References: chain due to limitations on field length. In Internet electronic mail, the use of Message-Id: and In-Reply-To: fields are optional and their format and nature is only loosely constrained when they are present. Subject: lines for both Usenet messages and Internet mail are allowed to contain arbitrary text, clients are inconsistent in their use of *Re:* tags, and manual editing of Subject: lines further confuses the issue. Furthermore, current approaches to threading are to some extent misconceived, as they rely upon rapidly changing conventions in software communication.

While user clients typically insert in messages structural information useful for recovering threads, inconsistencies between clients, loose standards, creative user behavior, and the subjective nature of conversation make current threading systems only partially successful, and the situation is unlikely to change.

One approach to dealing with the above situation is to try to force clients to follow tighter standards for specifying threads. However, such an approach does not appear practical in light of the increasing diversity of

clients and the growing interconnection of only partially compatible messaging systems. Tighter standards also do not help in recovering thread structure from archived messages, since deletion of fields such as In-Reply-To: by archiving and digestifying programs is common.

It is also not clear that threads should be identified with trees of reply links. The reply command is often used to avoid retyping a mail address, rather than to continue a conversation. Further, users will disagree about what is on-topic in a thread, and off-topic responses can easily spawn subdiscussions. Conversely, on-topic contributors to a discussion may simply send a message rather than using the reply command.

This suggests that the links desired for display in a threading interface, and which result in structures to be processed as a unit, are actually not objectively defined "pattern-matching" or "structural" links. The link desired to be captured is that of a response in an ongoing discourse. The fact that users are able to participate in online discussions, despite the in-adequacies of current threading software, suggests that most messages contain the contextual information to understand their place in an ongoing conversation. Thus it is at least possible that an automated system will be able to make use of this information as well to make this conversational structure explicit as a thread.

The role of cohesion or linking between the parts of a dialogue has been recognized. Language provides a variety of mechanisms for achieving this cohesion. One such mechanism is *lexical cohesion* and in particular lexical *repetition*, that is, the repeating of words in linked parts of a discourse.

The phenomenon of lexical repetition suggests that the

similarity of the vocabulary between two messages should be a powerful clue to whether a response relationship exists between them. Measuring the similarity of vocabulary between texts is, of course, a widely used strategy for finding texts with similar topic to a query. Indeed, similarity-based methods have been used to construct hypertexts linking documents or passages of documents on the basis of topic similarity.

Attempts have also been made to go beyond unlabeled linking to use similarity matching in detecting discourse relations. Hearst's *TextTiling* algorithm (see M.A. Hearst, "Multi-paragraph Segmentation of Expository Text," 32nd Annual Meeting of the Association for Computational Linguistics at Pp. 9-16, Las Cruces, NM June 27-30, 1994) uses vector space similarity to decompose a text into topically coherent segments. Also used is the graph structure of a network of raw similarity links to infer meta-links corresponding to discourse relations such as comparison and summarization (see J. Allan, "Automatic Hypertext Link Typing," Proceedings of Hypertext `96, 1996). These lines of evidence suggest text similarity could be a clue to the existence of a response relation between messages as well.

What is desired is a way to utilize robust conventions in human communication in place of, or in addition to, software conventions in order to produce an effective message threading system.

Summary of the Invention

An object of the present invention is to utilize the textual context and characteristics of messages to provide a more reliable and effective way to construct message

threads. In accordance with the present invention, statistical information retrieval techniques are used in conjunction with textual material obtained by "filtering" of messages to achieve a significant level of accuracy at identifying when one message is a reply to another.

Brief Description of the Drawings

FIGURE 1 shows the results of experimentation for a matching strategy used in an embodiment of the present invention.

FIGURE 2 contains a diagram showing an embodiment of the present invention.

FIGURE 3 contains a diagram showing a more generalized embodiment of the present invention.

Detailed Description

Threading of electronic messages should be treated as a language processing task. The present invention utilizes textual context and characteristics of messages in order to provide a more reliable and effective way to construct message threads. Preliminary experiments show that a significant level of threading effectiveness can be achieved by applying standard text matching methods from information retrieval techniques to the textual portions of messages. In accordance with the present invention, statistical information retrieval techniques are used in conjunction with textual material obtained by "filtering" of messages to achieve a significant level of accuracy at identifying when one message is a reply to another. A preferred embodiment of the present invention will now be described with reference to the experiments described below. The experiments are meant to be illustrative of the

process of the present invention and are not intended to be limiting.

Experiments

The goal in experimentation was to test the ability of various linguistic clues to indicate whether one message was a response to another. Three types of textual material from messages were investigated: (1) the Subject: line; (2) quoted material in the message; and (3) the (unquoted) text of the message itself. The results of the experiments conducted show that statistical information retrieval techniques can achieve a significant level of accuracy at identifying when one message is a reply to another.

Text from the Subject: line is a good clue that a message belongs to a particular thread, though it may not directly indicate which message in the thread is being replied to. Quoting of material from the parent message, particularly quotes of several lines, is a much stronger form of context. Salton and Buckley in an article entitled "Global Text Matching for Information Retrieval," Science, 253:1012-1015 (August, 1991), showed that text matching on a collection of Usenet messages which included substantial quoted material was highly effective at retrieving related messages, under a definition of relatedness that subsumed the response relationship of interest.

Further, the actual text of the reply can be expected, based on the coherence phenomena described earlier, to repeat words from the parent message. Since new material will be present as well, it is expected this to be a somewhat weaker clue than the Subject: line and quoted text.

a. Data Set and Preparation

A corpus of 2435 messages posted to the *www-talk* mailing list, during the period February 1994 through July 1994 were obtained from the archives at URL

http://www.w3.org/hypertext/WWW/Archive/www-talk.

A total of 941 of these messages had an In-Reply-To: field containing a unique identifier from the Message-Id: field of another message in the corpus. While it is suggested herein that In-Reply-To: links will not always correspond to the discourse response links of interest, they provide a reasonable initial test of the ability of text matching to find connections that are response-like. Therefore, these 941 child-parent pairs were used as ground truth against which methods for finding parent messages were tested.

Simple message filters were written to extract the three types of textual material (referred to above) from each message: (1) the text of the Subject: field; (2) unquoted text from the message body; and (3) quoted text from the message body. This resulted in three collections of 2435 document representatives, one for each type of textual material. Some messages had empty document representatives in some of the databases (for instance, a message might have no quoted material) and so could not be retrieved from that database. These messages were used as "target" messages for the matching strategies described herein. Target messages represented the potential parent messages matched against a given "query" (child) message chosen from the database. The "best" match of the target messages (excluding the query message) for a given query message represents a potential parent message.

Each of the three collections was indexed using Version 11.0 of the SMART experimental text retrieval system, obtained June 13, 1995 from directory *pub/smart* at *ftp.cs.cornell.edu*. The SMART text retrieval system uses statistical information retrieval techniques to rank target messages based using the cosine similarity formula and a

variant of *tf x idf* weighting. Using the SMART system, target messages were represented as vectors of numeric weights:

$$< w_{i1}, w_{i2}, \ldots w_{ik}, \ldots, w_{it} >$$

where

$$w_{ik} = \frac{f_{ik}}{\sqrt{\sum_{l=1}^{t} f_{il}^2}}$$

and f_{ik} *is* the number of times word k appears in message I. Query messages were similarly represented as vectors:

$$< q_1, q_2, \ldots q_k, \ldots, q_t >$$

where

$$q_k = \frac{f_k \, x \log(N/n_k)}{\sqrt{\sum_{l=1}^{t} (f_l \, x \log(N/n_l))^2}}$$

Here f_k is the number of times the word occurs in the query message, N is the number of messages in the database, and n_k is the number of messages containing word k. SMART scores each target message I as

$$\sum_{l=1}^{t} q_l w_{il}$$

b. Processing

Five text matching strategies were tested in the experiments for their ability to retrieve the parent of a

message, given text from the child message. For each strategy, all 941 document representatives of identified child messages were run as queries against one of the three databases of 2435 document representatives using the SMART system. This produced a ranking of all 2435 target (that is, potential parent) messages for each query message. Messages which did not have any words in common with the query were not retrieved. They were assigned random ranks lower than that of any retrieved message. Documents were ranked by the score assigned by the SMART system processing. The code developed for carrying out the processing, message filtering and matching (with the exception of the SMART program which, as noted, was obtained from a publicly-available source) is included in the appendix (pages 25 - 80), which is filed herewith and expressly incorporated by reference herein.

Each strategy was a choice of what text from a child should be used as a query (i.e., what type of message filter to use for a child message), and what text from target messages (i.e., what type of message filter) should be used to represent them in the database. The five combinations explored were:

Queries	Targets
Subject text	Subject text
Unquoted text	Unquoted text
Unquoted text	Quoted text
Quoted text	Unquoted text
Quoted text	Quoted text

c. Experimental results

FIGURE 1 displays the distribution of ranks of the 941 parent documents with respect to each of the five forms of text matching. The value for rank 0 is the number of times a child retrieved its parent as the first document in the ranking, rank 1 indicates how often the parent was second in the ranking, and so on. In computing the rank of the parent, the child document (which was itself present in the database, though not necessarily in the same form as was used in querying) was removed from the ranking, so that the ranks run from 0 to 2433 instead of 0 to 2434.

Table 1 below shows the number of times the parent was retrieved at rank 0, ranks 0 to 4, and ranks 0 to 9 for each of the search strategies used in the experimentation, over 941 trials. Comparison of this is made to the values that would be expected if the parent appeared at a random rank between 0 and 2433.

Ranks	Random	Subj-Subj	Unquot-Unquot	Unquot-Quot	Quot-Unquot	Quot-Quot
0	0.39	119	131	40	666	150
0-4	1.93	446	303	161	745	319
0-9	3.87	639	418	210	759	368

Table 1: Parents retrieved for each search strategy

Discussion

As expected, using the quoted portion of a message as a query (i.e., child message filter extracts quoted text portion) and matching against the unquoted portions of target messages (i.e., target message filter extracts

unquoted text) was the most effective strategy, of the five strategies tried, for finding a parent message. As shown in Table 1, the parent was the highest ranked message in 666 out of 941 trials or 71% of the time (for the quoted query - unquoted target strategy). Put another way, a system that simply assumed the highest ranked message under this matching strategy was the parent would, on average, have 0.71 recall (i.e., retrieval of 71% of the items relevant to the query message) and 0.71 precision (i.e., 71% of the retrieved items are relevant to the query message) at finding parent messages. Of course, these results are for messages that are known to have a parent message. An operational system would need not only to distinguish among potential parents, but also to detect whether or not the message has a parent at all. One way of accomplishing this is to establish a threshold -- which may be preset or specified by a user -- against which the ranking or similarity scores for the child and potential parent messages would be measured. If the highest ranking or similarity score falls below the threshold, then it would be determined that there is no "match", i.e., no true parent message for that child message.

These results can be roughly compared with the 0.90 recall and 0.72 precision in Salton and Buckley's experiments with Usenet messages containing quoted material. However, Salton and Buckley were attempting to find related messages, not just parent messages, and defined all messages with the same Subject: line as being related. The task undertaken by Salton and Buckley is a simpler task than finding the single parent of a message.

Referring again to FIGURE 1, it is apparent that the other strategies tried were not as effective as matching quoted text against unquoted targets, though all were far

better than random at finding parent messages. Even matching unquoted text queries against quoted text targets, which preferentially retrieves the children of a message, returns a nontrivial number of parents based on general content similarity. Similarly, quoted queries against quoted targets mostly should find siblings of a message, but gets some parents due to nested quotations that persist to the child.

How fast the number of parents gained drops off with increasing rank also depends on the matching strategy. As shown in FIGURE 1, the smoothest decay comes from matching unquoted material against unquoted material (the fourth curve in FIGURE 1). This picks up parents based on a general similarity of content rather than repetition of actual text from the parent. The relatively smooth gradation of content similarity which shows up in typical text retrieval systems also shows up here. In contrast, the curve for quoted queries vs. unquoted messages drops off extremely sharply. In most cases only the single parent messages will have a large block of unquoted text similar to the quoted text of the child. The curve for subject vs. subject (the fifth curve in FIGURE 1) drops sharply at the beginning, after the exhausting of those cases where there are nearly exact matches between the Subject: line of the query and a few documents with the same Subject: line. Later the curve is more gradual reflecting cases where the subject line is common to many messages, or the match is on only a subset of the words.

The diagram in FIGURE 2 shows the flow of message processing in accordance with the present invention. At 200 is a set of N target messages (denoted 1, 2, ..., N), any of which may be a parent message to be determined. Each target (potential parent) message at 200 is filtered

through a parent message filter A at 210. As seen from the experiments described above, parent message filter A may extract subject text, unquoted text, or quoted text from each message. The result of the message filtering operation is a set of filtered target (potential parent) messages (denoted 1_A, 2_A, ..., N_A) at 220. Preferably, based upon the above test results, message filter A at 210 extracts unquoted text from each potential parent, and the set of unquoted text messages for potential parents is at 220.

Continuing, the filtered potential parent messages (1_A, 2_A, ..., N_A) at 220 are then passed along to a Statistical Information Retrieval Function at 230. Statistical Information Retrieval Function 230 can be the SMART system described above or an equivalent statistically--based retrieval function.

The child, or reply, message CM at 240 is also processed using a message filter Q at 250. As discussed above, the child message filter may extract subject text, unquoted text, or quoted text from the child message, producing a filtered child message CM_Q at 260. Preferably, based upon the experiments described above, the child message filter at 250 extracts quoted text from the child message CM at 240, producing child quoted text at 260.

The filtered child message CM_Q is then passed to the Statistical Information Retrieval Function at 230, along with filtered parent messages (1_A, 2_A, ..., N_A). The Statistical Information Retrieval Function processes these message components to provide a similarity value table at 270, which represents values (denoted AQ_1, AQ_2, ..., AQ_N) each of which is a measure of how likely it is that the corresponding message (1, 2, ..., N) is the parent for the child message CM.

To determine the most likely parent message, the similarity value table at 270 is processed by a maximum value function at 280 from which the maximum value can be determined. The position in the table of the maximum value is a pointer or identifier at 290 that can be used to retrieve the corresponding target message which has been selected as the most likely parent message. This message can now be presented to the user along with the child message in a variety of formats, or simply retained for further processing to produce a thread. Alternatively, a list of potential message pairings -- with or without selecting which one is the actual parent -- may be presented to the user.

As mentioned above, an alternative step may include establishing a threshold against which the ranking or similarity scores for the child and potential parent messages are measured, and if none of the rankings or similarity scores exceed the threshold, then it would be determined that there is no "match", i.e., no true parent message for that child message.

Generating a thread may be accomplished by iteratively applying the method of the present invention as described above. Starting with a perceived child message, a likely parent message is determined using the method. That parent message is then substituted as a new "child" message and its parent (i.e., the grandparent of the original child message) is determined using the same method. Similarly, the grandparent message can then be substituted as yet another "child" message to determine its parent and so forth, so that ultimately a thread of messages having parent-child relationship between successive messages may be obtained.

Another way to generate a thread of messages is to

process all messages as child messages against all other messages as potential parent messages (which, in fact, is the technique utilized during experimentation). For each child message, its parent is determined as described above using a statistical information retrieval function and computing similarity values. Threads can be determined by linking up successive child-parent pairs. Linking of successive child-parent pairs may be done by, for example, finding a child message (denote as "B") having a parent message (denote as "A") wherein child message "B" is itself a parent message for another child message (denote as "C"); that is, message "A" is the parent of "B" and the grandparent of "C" Thus, the link of messages would be "A" - "B" - "C", and so on until all messages in the thread are accounted for.

An alternative to the embodiment of the present invention described above may be used to obtain a likely child message given a parent message. The basic process using message filters is the same for the alternative embodiment. The differences in the process are the filters used. For example, in the experiments described above, the best results in determining a parent message given a child message were obtained by using a quoted text filter for the child and an unquoted filter for each of the potential parent messages. Starting with a given parent message, then, the process would involve the use of an unquoted filter on the parent message and a quoted filter for each of the remaining messages (the potential child messages). Once the messages are filtered, the processing essentially takes place as described above.

It is readily apparent that one way of utilizing the present invention is with batch processing of messages such as, e.g., would be done in connection with message

archiving. Another way of utilizing the method of the present invention, however, is in the processing of incoming messages as they arrive, rather than waiting for a batch to accumulate. For example, when a new message arrives, the method of the present invention could be applied to identify a parent message from the messages that have previously arrived. In addition, in the event that the messages are received out of order, the new message could be checked against the other messages (in accordance with the method described above for locating a child message from a potential parent) in order to determine a child message for the newly received message.

A variety of improvements in the basic processing scheme described above are possible. By improving processing of document text, as well as making use of additional evidence, it is believed that the above results can be greatly improved. The improvements, each of which might be viewed as a message "filter," are as follows.

(1) Better Text Representation. The above-described experiments ignored the order of words when matching query messages against potential parents. This is sensible for detecting similarity of topic, as is the goal in matching unquoted text against unquoted text. A quotation in a child message, however, is likely to repeat a long sequence of words from the parent. Indexing, matching, and term weighting based on multi-word phrases or entire lines should greatly reduce the number and strength of spurious matches. Since header material (From: lines, etc.) can appear in quotes as well, matching should be allowed on this material as well.

(2) Nested Quotation. Multiple levels of quotation are common in electronic messaging, and are indicated by concatenated prefixes. For instance, if textual material

is prefixed by ">> >", it would be expected that the parent message has the material prefixed by "> >", or perhaps by ">", but probably not by nothing and certainly not by "|" or "*". Concatenated *Re:* tags appear in **Subject:** lines, but should be statistically characterized, since their use by mailers is erratic.

(3) Time. Most replies to a message occur within a window of a few days after the message is posted. A simple statistical model, perhaps similar to those used in analyzing citation patterns, can be used to take this tendency into account.

(4) Recognizing Other Message Relationships. Duplicated, bounced, reposted, continued, and revised messages have strong textual similarity to other messages. The experimental data showed cases where they were falsely construed as replies. If treated simply as nonreplies they are likely to distort statistical models distinguishing replies from nonreplies. A better approach is to model these other message relationships as well, both to distinguish them from response relationships and to provide additional useful links between messages. For instance, a mail reader might display a revised message while backgrounding the original.

(5) Authorship Information. Replies often refer to the author of the parent message, either in an automatically produced fashion (such as):

lewis@research.att.com (David L. Lewis) writes:

>I'd really like a threading email reader.

or via a manually written salutation (e.g., *Dear Susan*). These may be matched against header information of messages

and manually or automatically produced signatures.

(6) Cue Phrases. In responses which do not directly quote the parent message, the author will often use linguistic cues to indicate the parent message, e.g. I *really like the suggestion that...* or *Your argument is....* Considerable research which has been done on distinguishing what relationship a particular cue phrase is indicating can be applied.

(7) Message Categorization. Certain types of messages such as calls for papers and job ads are unlikely to be replies to other messages and/or are unlikely to be replied to publicly. Known text categorization methods can detect these and provide evidence against the presence of response links.

(8) Detection of Siblings. A message without a clear connection to its parent may be similar to another child of the same parent, which does have a clear like. For instance, two people may post similar responses objecting to an error in the parent message, but only one uses the reply command.

All of the above improvements are, in effect, clues that provide evidence toward the presence or absence of response links, but in all cases this evidence is uncertain. A planned strategy is to implement the clues so as to reduce their uncertainty as much is as reasonable, but then to rely on machine learning methods known to those skilled in the art to combine these multiple uncertain clues into a decision procedure. This approach to complex information retrieval problems allows the system implementer to focus on the relatively clean task of building feature detectors, while letting a learning algorithm use training data to balance the uncertain relationship of those features to the property of interest.

(Two articles provide good examples of this strategy: B. Croft, J. Callan & J. Broglio, "Trec-2 routing and Ad-hoc Retrieval Evaluation Using the Inquery System," in The Second Text Retrieval Conference (D.K. Harman, ed., Gaithersburg, MD, March 1994, U.S. Dept. of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-215) pp. 75-83; and E. Spertus, "Smokey: Automatic Flame Recognition," Manuscript, Computer Science Department, Massachusetts Institute of Technology, 1996, submitted to ACM SIGIR `96.) In addition, this approach allows the system to be tailored to user preferences as expressed, for instance, through their overriding of system decisions. This is desirable, since the presence of a response link is to some degree subjective.

Each of the above-referenced improvements may be utilized as message filters alone or in combinations with one another and with the "subject text," "quoted text" and "unquoted text" message filters that were the subject of the experiments described herein. Accordingly, an embodiment of the present invention may be obtained as a generalization of the embodiment reflected in FIGURE 2 described above. With reference to the diagram in FIGURE 3, the flow of message processing for the more general embodiment of the present invention will now be described.

As shown in FIGURE 3, at 300 is a set of N target messages (denoted 1, 2, ..., N), any of which may be a parent message to be determined. Each target (potential parent) message at 300 is filtered through a parent message filter bank (which may be one or more message filters). The parent message filter bank is shown at 310 in FIGURE 3 as a set of one or more message filters denoted by A, B, ..., K, giving a parent message filter bank of length K.

Parent message filters A through K may extract subject text, unquoted text, or quoted text from each message, or they may implement one or more of the "improvements" in message analysis described above (such as, e.g., extracting nested quotations, time information, or cue phrases). The result of the filtering operation is a set of N filtered target (potential parent) message vectors (denoted 1_A, 1_B, ..., 1_K, 2_A, 2_B, ..., 2_K, ..., N_A, N_B, ..., N_K) at 320, where each filtered parent message is a vector consisting of the K filtered representations of the message, i.e., each element of the vector is the result of one of the K filtering operations (e.g., filtered target message 1 is denoted as vector 1_A, 1_B, ..., 1_K, where 1_A represents the result of processing target message 1 through message filter A, etc.). These filtered potential parent messages at 320 are then passed along to Statistical Information Retrieval Function at 330, which may be the SMART system described above or an equivalent statistically--based retrieval function.

The child, or reply, message CM at 340 is also processed using a message filter bank (which may be one or more message filters). In FIGURE 3, the child message filter bank is shown at 350 as a set of message filters denoted as Q, R, ..., Z, giving a child message filter bank of length Z-Q+1. The child message filter bank may contain one or more of the same type of potential message filters described above for the parent message filter bank. The child message filter bank produces a filtered child message vector (denoted CM_Q, CM_R, ..., CM_Z) containing Z-Q+1 filtered representations of the message at 360.

The filtered child message vector (CM_Q, CM_R, ..., CM_Z) is then passed to the Statistical Information Retrieval Function at 330, along with the set of filtered parent

message vectors (1_A, 1_B, ..., 1_K, 2_A, 2_B, ..., 2_K, ..., N_A, N_B, ..., N_K). The Statistical Information Retrieval Function processes these message components to provide a similarity value table at 370, with values (denoted AQ_1, AQ_2, ..., AQ_N, KZ_1, KZ_2, ..., KZ_N) representative of the similarity between potential parent and child message components. It may be preferable to combine the columns of values in the similarity value table of 370 using a combiner function at 372 to provide a single tuple of values at 374, each element of which is a measure of how likely it is that the corresponding message (1, 2, ..., N) is the parent for the child message CM. As discussed above, the combiner function may be a decision procedure based upon machine learning methods. To determine the most likely parent message, the tuple of values at 374 is processed by a selector function at 380 from which an identifier for the most likely parent message can be determined at 390. For example, if the selector function is the maximum value function described above with reference to FIGURE 2, the position of the maximum value in the tuple of values is a pointer or identifier at 390 that can be used to retrieve the corresponding target message which has been selected as the most likely parent message. The selected message can now be presented to the user along with the child message in a variety of formats, or simply retained for further processing to produce a thread.

Those skilled in the art will recognize that in the latter-described embodiment of present invention, each of the parent and child message filter banks may consist of a single message filter or multiple message filters. Those skilled in the art will further appreciate that the present invention may be implemented in any one of a number of known ways. For example, the present invention may be

implemented by integrating or combining the techniques of the present invention with an e-mail reader or browser software program. Such a program may be client-based (i.e., found locally within an individual's personal computer) or server based (i.e., found in a computer or gateway remote from the individual reader). As another example, the present invention could be implemented as part of a client-based or server-based message archival software program. The advantages of the present invention do not depend upon the particular mode of operation (i.e., server or client) of a computer or processor through which the techniques herein described are implemented. It will be clear to those skilled in the art that the location of the messages that may be processed in accordance with the invention described herein need not be stored in the same location as the program utilized for carrying out such processing. Indeed, messages may be downloaded to a client station or to a message server from a remote location, such as, e.g., a message database accessible over the Internet or accessible over a corporate intranet.

In summary, instead of attempting to solve the e-mail threading problem by forcing more consistency in the use of structural links by client software, the present invention involves an approach to threading that makes use of a range of individually uncertain, but cumulatively compelling clues as to what is going on in a conversation.

What has been described is merely illustrative of the application of the principles of the present invention. Other arrangements and methods can be implemented by those skilled in the art without departing from the spirit and scope of the present invention.

DIRNOT.TXT Fri May 2 09:32:23 1997 1

29-may-96 DDL : Copies of all the following files were sent to Gene Nelson today. for the preliminary patent application.

```
arran $ cd ~/projects/Aactive/emailpatent/dircopies/project/
arran $ ls -lR
total 12
drwxrwxrwx  6 kknowles     4096 May 29 09:01 bin
drwxrwxrwx  2 kknowles     4096 May 29 09:56 explorations
drwxrwxrwx  2 kknowles     4096 May 28 14:09 make_files

bin:
total 20
-rwxrwxrwx  1 lewis        1556 May 29 08:57 README2
drwxr-xr-x  2 lewis        4096 May 29 09:44 dirformat
drwxr-xr-x  2 lewis        4096 May 29 09:01 dirqueries
drwxr-xr-x  2 lewis        4096 May 29 09:01 dirscores
drwxr-xr-x  2 lewis        4096 May 29 09:01 dirutil

bin/dirformat:
total 96
-rw-rw-rw-  1 lewis        6137 May 29 08:57 README2-formatting
-rw-r--r--  1 lewis       30077 May 29 09:44 all2.ps
-rwxrwxrwx  1 kknowles       115 Jul 21  1995 get-body
-rwxrwxrwx  1 kknowles       896 Jul 24  1995 get-quoted
-rwxrwxrwx  1 kknowles       607 Aug  1  1995 get-unquoted
-rwxrwxrwx  1 kknowles       607 Aug 23  1995 get-unquoted-digest
-rwxrwx-wx  1 kknowles      1083 Aug 23  1995 headerparser
-rwxrwx-wx  1 kknowles       475 Jul 17  1995 mailbody1
-rwxrwx-wx  1 kknowles       529 Jun 28  1995 mailbody2
-rwxrwxrwx  1 kknowles       791 Jul 31  1995 make-quoted-files
-rwxrwxrwx  1 kknowles       715 Jul 31  1995 make-subj-files
-rwxrwxrwx  1 kknowles       992 Aug 23  1995 make-unquoted-digest-files
-rwxrwxrwx  1 kknowles       978 Aug  1  1995 make-unquoted-files
-rwxrwxrwx  1 kknowles       644 Aug 23  1995 parse-digest
-rwxrwx-wx  1 kknowles      2495 Aug 23  1995 parse-messages
-rwxrwxrwx  1 kknowles       693 Aug 22  1995 trim-digest

bin/dirqueries:
total 24
-rw-rw-rw-  1 lewis        2653 May 29 08:57 README2-runningqueries
-rwxrwxrwx  1 kknowles     1453 Aug 22  1995 query-irts
-rwxrwxrwx  1 lewis         856 May 28 13:34 query-quoted
-rwxrwxrwx  1 kknowles     1101 Aug 23  1995 query-run
-rwxrwxrwx  1 kknowles      575 Aug 23  1995 query-run-better
-rwxrwxrwx  1 kknowles      804 Aug  7  1995 runsmart

bin/dirscores:
total 64
-rw-rw-rw-  1 lewis        5353 May 29 08:57 README2-computingscores
-rwxrwxrwx  1 kknowles      802 Aug 22  1995 child-parent-rank-score
-rw-rw-rw-  1 kknowles     5210 Aug 25  1995 create_full_files
-rwxrwxrwx  1 kknowles     1915 Aug 23  1995 finish-cprs
-rwxrwx-wx  1 kknowles     1332 Jul 24  1995 get-score
-rwxrwxrwx  1 kknowles     1513 Aug 23  1995 kimjoin
-rwxrwxrwx  1 kknowles     2105 Aug 21  1995 make-score
-rwxrwx-wx  1 kknowles      218 Aug 16  1995 make-score-list
-rwxrwxrwx  1 kknowles      465 Aug 21  1995 num-docs-retrieved
-rwxrwx-wx  1 lewis         530 May 28 13:44 smartbatch1
-rwxrwxrwx  1 kknowles     2075 Jul 25  1995 test-children
-rwxrwxrwx  1 kknowles     4753 Jul 27  1995 test-irt
-rwxrwxrwx  1 kknowles     1788 Aug 16  1995 test-irt-better

bin/dirutil:
total 20
-rw-rw-rw-  1 lewis        1013 May 29 08:57 README2-utils
-rwxrwxrwx  1 kknowles      405 Aug 23  1995 forall
-rwxrwxrwx  1 kknowles      622 Aug 14  1995 geomean
-rwxrwx-wx  1 kknowles      208 Aug  4  1995 rand-selector
-rwxrwxrwx  1 kknowles      886 Jul 21  1995 stdevchildparent

explorations:
total 32
-rw-rw-rw-  1 lewis       14758 May 28 14:06 004-notes-using-smart-to-find-parents
-rw-rw-rw-  1 lewis       13480 May 28 14:07 008-preparing-data-sets-digest-format

make_files:
total 12
-rwxrwxrwx  1 lewis        2009 May 28 14:08 make_text_quot
-rwxrwxrwx  1 lewis        1889 May 28 14:08 make_text_subj
-rwxrwxrwx  1 lewis        1991 May 28 14:08 make_text_uquot
arran $
```

David D. Lewis and Kimberly Knowles, 29-May-96

This directory contains all email threading software used in producing the results in:

```
@article(LEWIS96e
, author       = ``David D. Lewis and Kimberly A. Knowles''
, title        = ``Threading Electronic Mail: A Preliminary Study''
, journal      = ``Submitted for publication. Comments welcome.''
, year         = 1996
, copyrightown = ``AT&T 1996''
)
```

The only exception is the code of the SMART system, a publically available text retrieval system described in:

```
@book(SALTON83
, author = "Gerard Salton and Michael J. McGill"
, title = "Introduction to Modern Information Retrieval"
, publisher = "McGraw-Hill Book Company"
, year = 1983
, address = "New York"
 )
```

The capabilities of the SMART system are very similar to those of a number of commercial text retrieval systems, such as SearchServer from Fulcrum, Inc. of Ottawa, Canada.

The files are broken up into 4 groups, as described in these four files:

README2-formatting : Formatting the raw documents, including breaking up files of messages into individual messages, and pulling out quoted, unquoted, and subject line material.

README2-runninqqueries: Using various parts of potential child documents as queries to be matched against databases of documents.

README2-computingscores: Combining the results of the multiple queries in different ways to get rankings and scores.

README2-utils: Miscellaneous utility functions.

In this printout, the text of each of these files is followed by the source code for that part of the software.

README2-formatting Fri May 2 09:15:53 1997 1

David D. Lewis and Kimberly Knowles, 29-May-96

---

get-body: 5/21/95 runs mailbody1 for all files in dir.

---

mailbody1: 7/17/95 strips out only the body part of a message. called by smartpatch1.

---

mailbody2: 6/28/95 does the same for html (hypermail) formatted messages.

---

get-quoted: 7/17/95 mailbody1 for quoted text only. a filter which outputs only the lines of arg which begin with any number of types of prefixes, and crop that part. called by query-quoted to run smart on quoted material. called by make-quoted-files to make dir of files of quoted material.
Let through just the quoted body part of a mail message
in mbox format. (Only one message in file.)
called on by query-quoted
#
What I really want is to be able to denote a list of possible prefixes somewhere,
and just say if any of these, then strip them and print. otherwise, skip it over.
how about if (/pattern1/ || /pattern2/ || ... || /patternN/) { s/\1//; print;} ?

---

get-unquoted: 8/1.95 does the same as get-quoted but for unquoted (nonquoted) text only. called by make-unquoted-files.
Let through just unquoted part of the body of a mail message
in mbox format. (Only one message in file.)

---

get-unquoted-digest. 8/23/95 does the same as get-unquoted but uses "Date: " as a message delimiter. instead of "From ".

---

headerparser: 6/27/95 removes listserv to kim headers from archive files which were mailed in digest format. removes all pinhead headers.
Usage.
headerparser FILENAME DESTINATION
#
This is to parse messages of archives that I receive from a
listserver. the header from the server to me is what I want
to get rid of. the archives, in digest format, can be dealt
with separately or by expanding the program eventually.
Note we know the first line is a 'From ' line.
Note also that the initial header is followed by a blank line.
So if we look for a blank line after the 'From ' line, we can
delete up to there and continue. In fact, since the rest is
in digest format, we know that 'From ' must signal the header.

---

make-quoted-files: 7/31/95 makes a directory of files. calls on get-quoted to filter for quoted material only.
Usage: make-quoted-files DIR DEST
where DIR is the source directory and DEST is the directory where you want the files put.
#
this will create files of just quoted material contents
#
#for some reason, it is creating an empty dotfile. this will screw up the message id's
for smart, so be sure to remove it. i'm not really sure why it's happening.

---

make-subj-files: 7/29/95 prints subject line of all files in dir to outdir, one file per subject. a direct mapping of message to subject file.
Usage: make-subj-files DIR DEST
#
this will create files of just subject contents, with Re: removed if
it is there.
#for some reason, it is creating an empty dotfile. this will screw up the message id's
for smart, so be sure to remove it. i'm not really sure why it's happening.

---

make-unquoted-digest-files: 8/23/95 same as make-unquoted-files, only it calls on get-unquoted-digest.
Usage: make-unquoted-digest-files DIR DEST
where DIR is the source directory and DEST is the directory where you want the files put.
#
this will create files of just unquoted material contents.
#
#this was copied directly from make-quoted-files, with a 3-character change. I guess -
if I were doing this a lot more times, I'd just take the filter script as a parameter.
#
#for some reason, it is creating an empty dotfile. this will screw up the message id's
for smart, so be sure to remove it. i'm not really sure why it's happening.

```
-----------------------------------------
make-unquoted-files: 7/31/95 prints unquoted text of files to outdir, one file per
subject.
# Usage: make-unquoted-files DIR DEST
# where DIR is the source directory and DEST is the directory where you want the files
# put.
#
# this will create files of just unquoted material contents.
#
#this was copied directly from make-quoted-files, with a 1-character change.  I guess
# if I were doing this a lot more times, I'd just take the filter script as a
# parameter.
#
#for some reason, it is creating an empty dotfile.  this will screw up the message id's
# for smart, so be sure to remove it.  I'm not really sure why it's happening.

-----------------------------------------
parse-digest: 8/22/95 parses messages in digest format, using "Date: " as a
delimiter
# Usage  parse-digest infile dir
#
# This is a perl script meant to parse messages in digest format into
# separate files, one message per file.
#
# This will work similarly to parse-messages.
#
# Messages are begun with the digest header "Date: "

-----------------------------------------
parse-messages: 7/9/95 uses pinhead to cut messages into indiv files called
000000.mbox.  takes archive file as argument, destination file and number to
append to are hardwired, well not any more.
# Usage: parse-messages OUTDIR LIST OF ARCHIVE FILES
#
# This is a perl script meant to parse messages in .mbox format into
# separate files, one message per file.
#
# This will work similarly to
# /home/lewis/projects/Aactive/email/bin/dlbreak7
# but I don't think it will have to sort by mailing list, since the
# archives are already in the same file by mailing list.
#
# Messages are begun with the pinhead header "From <stuff> <Date>" or
# some approximation.

-----------------------------------------
trim-digest: 8/22/95 designed to take the separater string out of parsed messages
from digest archives.
# usage: trim-digest messagedir
#
# to trim out the last line of digest files parsed using parse-digest.
# I should just encapsulate them in the future.
```

get-body Fri May 2 09:15:53 1997 1

```
#!/bin/ksh
#
#
prefix=$1

for file in ${prefix}'*
do
  /home/kknowles/project/bin/mailbody1 $file
done

exit 0
```

get-quoted Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Let through just the quoted body part of a mail message
# in mbox format. (Only one message in file.)
# called on by query-quoted.
#
# What i really want is to be able to denote a list of possible prefixes somewhere,
# and just say if any of these, then strip them and print. otherwise, skip it over.
# how about if (/pattern1/ || /pattern2/ || ... || /patternN/) { s/\1//; print;} ?

#$pat1 = "([A-Za-z]+>)";
$pat2 = "([A-Za-z]*>+)";
$pat3 = "(\\*)";
$pat4 = "([A-Za-z]*\\|+)";
#$pat = "$pat1|$pat2|$pat3|$pat4";
$pat = "$pat2|$pat3|$pat4";

while (<>) {
    if (/^(($pat)($pat|(\s))*)/) {
        s/\1//;                    # will this work?
        print;
    }
#    elsif (/^[   ]/) {  # this doesn't work; meant to filter the quotations
# set off by indents.  message 76 has 3 spaces indentations, plus later on it has
# code indented (that's not a quote).  sigh.
#        s/\1//;
#        print;
#    }
}
```

get-unquoted Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Let through just unquoted part of the body of a mail message
# in mbox format. (Only one message in file.)
#
$pat2 = "([A-Za-z]*>+)";
$pat3 = "(\\*|";
$pat4 = "([A-Za-z]*\\|+)";
$pat = "$pat2|$pat3|$pat4";

$state=0;
while (<>) {
  if ($state == 0) {
    if (/^From /)
      {$state=1;}
    elsif (/\S/)
      {die("Non-blank in line before From . in line $.");}
  }
  elsif ($state == 1) {
    if (/^\s*$/)
      {$state=2;}
  }
  elsif ($state == 2) {
    if (! /^.$/)
      {print unless (/^([!\s*)($pat)($pat|(\s))*)/) ;}
  }
  else
    {die("Should not get here.");}
}

#END
```

get-unquoted-digest Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Let through just unquoted part of the body of a mail message
# in mbox format. (Only one message in file.)
#
$pat2 = "([A-Za-z]*>+)";
$pat3 = "(\\*)";
$pat4 = "([A-Za-z]*\\|+)";
$pat = "$pat2|$pat3|$pat4";

$state=0;
while (<>) {
  if ($state == 0) {
    if (/^Date: /)
      {$state=1;}
    elsif (/\S/)
      {die("Non-blank in line before Date: in line $.");}
  }
  elsif ($state == 1) {
    if (/^\s*$/)
      {$state=2;}
  }
  elsif ($state == 2) {
    if (! /^.$/)
      {print unless (/^((\s*)($pat)($pat|(\s))*)/) ;}
  }
  else
    {die("Should not get here.");}
}

#END
```

headerparser Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Usage:
# headerparser FILENAME DESTINATION
#
# This is to parse messages of archives that i receive from a
# listserver.  the header from the server to me is what i want
# to get rid of.  the archives, in digest format, can be dealt
# with separately or by expanding the program eventually.

# Note we know the first line is a 'From ' line.
# Note also that the initial header is followed by a blank line.
# So if we look for a blank line after the 'From ' line, we can
# delete up to there and continue.  In fact, since the rest is
# in digest format, we know that 'From ' must signal the header.

$archive=$ARGV[0];
$parsedarchive=$ARGV[1];
open(INFILE, $archive) || die "Couldn't open infile";
open(OUTFILE, ">$parsedarchive") || die "Couldn't open outfile";
$state=0;
while (<INFILE>) {
    if ($state == 0) {
      if (/^From /) {
        $state=1;
      }
      if ($state == 0) {
        print OUTFILE;
      }
    }
    elsif ($state == 1) {
      if (/^\s$/) {
        $state=0;
      }
    }
    else {
      die ("Shouldn't get here");
    }
}
```

mailbody1 Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Let through just the body part of a mail message
# in mbox format. (Only one message in file.)
# called on by smartbatch1.
#
$state=0;
while (<>) {
  if ($state == 0) {
    if (/^From /)
      {$state=1;}
    elsif (/\S/)
      {die("Non-blank in line before From .");}
  }
  elsif ($state == 1) {
    if (/^\s*$/)
      {$state=2;}
  }
  elsif ($state == 2) {
    if (! /^.$/)
      {print;}
  }
  else
    {die("Should not get here.");}
}

#END
```

mailbody2 Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Let through just the body part of a mail message
# in html format. (Only one message in file.)
#
$state=0;
while (<>) {
  if ($state == 0) {
    if (/^<\!-- body="start" -->/)
      {$state=1;}
  }
  elsif ($state == 1) {
      if (! /<p>\s*/) {
          if (/^<\!-- body="end" -->/) {
              $state=0;}
          else {
              chop;
              chop;
              chop;
              chop;
              chop;
              print "$_\n";}
  }
#  elsif ($state == 2) {
#      print;
#      $state=1;
#  }
  else
    {die("Should not get here.");}
}
```

make-quoted-files Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Usage. make-quoted-files DIR DEST
# where DIR is the source directory and DEST is the directory where you want the files
# put.
#
# this will create files of just quoted material contents.
#
#for some reason, it is creating an empty dotfile.  this will screw up the message id's
# for smart, so be sure to remove it.  i'm not really sure why it's happening.


#$/="";
#$*=1;

opendir(INDIR,$ARGV[0]);
@infile=readdir(INDIR);
closedir(INDIR);

$quotdir=$ARGV[1];

mkdir($quotdir,0755);

foreach (@infile) {
    open(MSGFILE, $ARGV[0] . $_);
    /(0\d+).mbox/;
    $flname=$1;
    open(OUTFILE, ">$quotdir$flname.quot");
    open(QUOTE, "get-quoted $ARGV[0]$_ |");
    while (<QUOTE>) {
        print OUTFILE $_;
    }
    close(QUOTE);
    close(OUTFILE);
    close(MSGFILE);
}
```

```
#!/usr/sbin/perl
#
# Usage: make-subj-files DIR DEST
#
# this will create files of just subject contents, with Re: removed if
# it is there.
#for some reason, it is creating an empty dotfile.  this will screw up the message id's
# for smart, so be sure to remove it.  i'm not really sure why it's happening.

$/="";
$*=1;

opendir(INDIR,$ARGV[0]);
@infile=readdir(INDIR);
closedir(INDIR);

$subjdir=$ARGV[1];

mkdir($subjdir,0755);

foreach (@infile) {
    open(MSGFILE, $ARGV[0] . $_);
    /(0\d+).mbox/;
    $flname=$1;
    open(OUTFILE, ">$subjdir$flname.subj");
    while (<MSGFILE>) {
        if (/^Subject: (Re: )?( *)/) {
            $subj=$2;
            print OUTFILE "$2\n";
        }
    }
    close(OUTFILE);
    close(MSGFILE);
}
```

make-unquoted-digest-files Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Usage: make-unquoted-digest-files DIR DEST
# where DIR is the source directory and DEST is the directory where you want the files
# put.
#
# this will create files of just unquoted material contents.
#
#this was copied directly from make-quoted-files, with a 3-character change.  i guess
# if i were doing this a lot more times, i'd just take the filter script as a
# parameter.
#
#for some reason, it is creating an empty dotfile.  this will screw up the message id's
# for smart, so be sure to remove it.  i'm not really sure why it's happening.


#$/="";
#$*=1;

opendir(INDIR,$ARGV[0]);
@infile=readdir(INDIR);
closedir(INDIR);

$quotdir=$ARGV[1];

mkdir($quotdir,0755);

foreach (@infile) {
    open(MSGFILE, $ARGV[0] . $_);
    /(\d+).mbox/;
    $flname=$1;
    open(OUTFILE, ">$quotdir$flname.uquot");
    open(QUOTE, "get-unquoted-digest $ARGV[0]$_ |");
    while (<QUOTE>) {
        print OUTFILE $_;
    }
    close(QUOTE);
    close(OUTFILE);
    close(MSGFILE);
}
```

make-unquoted-files Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Usage: make-unquoted-files DIR DEST
# where DIR is the source directory and DEST is the directory where you want the files
# put.
#
# this will create files of just unquoted material contents.
#
#this was copied directly from make-quoted-files, with a 3-character change.  i guess
# if i were doing this a lot more times, i'd just take the filter script as a
# parameter.
#
#for some reason, it is creating an empty dotfile.  this will screw up the message id's
# for smart, so be sure to remove it.  i'm not really sure why it's happening.


#$/="";
#$*=1;

opendir(INDIR,$ARGV[0]);
@infile=readdir(INDIR);
closedir(INDIR);

$quotdir=$ARGV[1];

mkdir($quotdir,0755);

foreach (@infile) {
    open(MSGFILE, $ARGV[0] . $_);
    /(0\d+).mbox/;
    $flname=$1;
    open(OUTFILE, ">$quotdir$flname.uquot");
    open(QUOTE, "get-unquoted $ARGV[0]$_ |");
    while (<QUOTE>) {
        print OUTFILE $_;
    }
    close(QUOTE);
    close(OUTFILE);
    close(MSGFILE);
}
```

parse-digest Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Usage: parse-digest infile dir
#
# This is a perl script meant to parse messages in digest format into
# separate files, one message per file.
#
# This will work similarly to parse-messages.
#
# Messages are begun with the digest header "Date: "
#

# We will be using strings with multiple lines.

$*=1;

#messages that you have.  the first one written to will be 1+ this number.
$num="00000";

$indir=$ARGV[0];
$outdir=$ARGV[1];

open (INFILE, $indir).

while (<INFILE>) {
    if (/^Date:/) {
        $num++;
        open(OUT_ARCHIVE,       ">$outdir/$num" . ".mbox") || die "couldn't open file";
    }
    print (OUT_ARCHIVE
}
```

parse-messages Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# Usage: parse-messages OUTDIR LIST OF ARCHIVE FILES
#
# This is a perl script meant to parse messages in .mbox format into
# separate files, one message per file.
#
# This will work similarly to
# /home/lewis/projects/Aactive/email/bin/dibreak7
# but I don't think it will have to sort by mailing list, since the
# archives are already in the same file by mailing list.
#
# Messages are begun with the pinhead header "From <stuff> <Date>" or
# some approximation.
#

# We will be using strings with multiple lines.
#!/usr/sbin/perl
#
$*=1;

#messages that you have.  the first one written to will be 1+ this number.
$num="00000";

#$outdir="/radish/070/projtrain/messages";

#/radish/070/netlists/archives/wwwtalk/WWW_Talk_1991.archive
#/radish/070/netlists/archives/wwwtalk/WWW_Talk_1992.archive
#/radish/070/netlists/archives/wwwtalk/WWW_Talk_1993q1.archive
#/radish/070/netlists/archives/wwwtalk/WWW_Talk_1993q2.archive
#/radish/070/netlists/archives/wwwtalk/www-talk-9402
#/radish/070/netlists/archives/wwwtalk/www-talk-9403
#/radish/070/netlists/archives/wwwtalk/www-talk-9404
#/radish/070/netlists/archives/wwwtalk/www-talk-9405
#/radish/070/netlists/archives/wwwtalk/www-talk-9406
#/radish/070/netlists/archives/wwwtalk/www-talk-9407

#@archives=("/radish/070/netlists/archives/wwwtalk/www-talk-9402","/radish/070/netlists/archives/wwwtalk/www-talk-9403","/radish/070/n
etlists/archives/wwwtalk/www-talk-9404","/radish/070/netlists/archives/wwwtalk/www-talk-9405","/radish/070/netlists/archives/wwwtalk/w
ww-talk-9406","/radish/070/netlists/archives/wwwtalk/www-talk-9407");
#@archives=("/radish/070/netlists/archives/wwwtalk/www-talk-9408","/radish/070/netlists/archives/wwwtalk/www-talk-9409","/radish/070/n
etlists/archives/wwwtalk/www-talk-9410","/radish/070/netlists/archives/wwwtalk/www-talk-9411","/radish/070/netlists/archives/wwwtalk/w
ww-talk-9412","/radish/070/netlists/archives/wwwtalk/www-talk-9501");
# this was just to test:
#@archives=("/radish/070/netlists/archives/wwwtalk/www-talk-9408");

# I think this will work.
$outdir=shift(@ARGV);
@archives=@ARGV;

#open (INFILE, "/radish/070/netlists/archives/smartpeople/smartpeople9495.mbox");
#open (INFILE, "/radish/070/netlists/archives/wwwtalk/www-talk-9410");

#while (<INFILE>) {

foreach (@archives) {
    open(INFILE, $_);
    while (<INFILE>) {
        if (/^From /) {
            $num++;
            open(OUT_ARCHIVE,       ">$outdir/$num    .mbox") || die "couldn't open file";
        }
        print (OUT_ARCHIVE);
    }
}
```

trim-digest Fri May 2 09:15:53 1997 1

```
#!/usr/sbin/perl
#
# usage: trim-digest messagedir
#
# to trim out the last line of digest files parsed using parse-digest.
# i should just encapsulate them in the future.

$thedir=$ARGV[0];

opendir(INDIR, $thedir) || die "Couldn't open indir";
@messages=readdir(INDIR);
closedir(INDIR);
shift(@messages);
shift(@messages);

foreach (@messages) {
    `mv $thedir/$_ tmp`;
    open(MSG,"tmp") || die "Couldn't open infile: $!";
    open(OUT,">$thedir/$_") || die "Couldn't open outfile";
    $b=<MSG>;
    $pline=$b;
    while ($a=<MSG>) {
        print OUT $pline;
        $pline=$a;
    }
    if ($pline =~ /^=+$/) {
    }
    else {
        print OUT $pline;
    }
    close(MSG);
    close(OUT);
}

`rm tmp`;
```

README2-runningqueries Fri May 2 09:17:12 1997 1

```
David D. Lewis and Kimberly Knowles. 29-May-96

-----------------------------------------
query-irts: 8/7/95 takes a filename of index, directory of files, specfile, and outfile as
arguments, runs smart using each file in the dir as a query, and prints it to
outfile.  num_wanted is a variable in the script, and can be changed appropriately.
#!/usr/sbin/perl
#
# Usage: query-irts keydir dirquery spec outfile
#
# first, make an array of the childids we are interested in, taken from file
# keydir (first column)
# then, run smart for each of them.
# store it in outfile.
# we should input a directory for the query files, and the spec file.
# SARGV[0]=file of child-parent pairs by message number
# SARGV[1]=directory of single file queries, string ending with '/'; must contain
# /subjects/ , /quotes/ , or /unquotes/
# SARGV[2]=location of corresponding spec file
# SARGV[3]=where to put the output
   NOTE: YOU SPECIFY TYPE OF QUERY BY GIVING IT A DIRECTORY



-----------------------------------------
query-quoted: 7/17/95 smartbatch1 for only running quoted material as a query.
calls on get-quoted.
# query-quoted PREFIX SPEC NUMOUT > OUT
#
# Description. This program is a demonstration of operating
# smart in batch mode.  Iterate over several documents (one
# per file), run them as smart queries, and save the
# results for the top 10 documents.
#
# history: adapted from smartbatch1
   NOTE: HARDCODED TO USE QUOTED MATERIAL AS QUERIES


-----------------------------------------
query-run: 8/1/95 like smartbatch, only it doesn't filter the document as it
comes in, it assumes the document that comes in is the one we want.
# Usage: query-run PREFIX SPEC NUMOUT > OUT
#
# Description: This program is a demonstration of operating
# smart in batch mode.  Iterate over several documents (one
# per file), run them as smart queries, and save the
# results for the top 10 documents.
#
# This version runs on each file, in its entirety.  therefore, it is meant for use when the
# files are preparsed, and the entire file is used as a query.  This removes the need to
# filter out the query by piping it from another script.


-----------------------------------------
query-run-better: 8/23/95 same as query-run, only it takes more as inputs and
doesn't require a pipe to output.
# Usage: query-run-better indir spec numout outfile
#
# similar to query-run; runs smart on each doc in indir across docs
# in spec.
$indir=$ARGV[0];
$spec=$ARGV[1];
$num_wanted=$ARGV[2];
$outfile=$ARGV[3];

-----------------------------------------
runsmart: 8/7/95 runs a file in smart, given file, spec, numout
# runsmart FILE SPEC NUMOUT
```

query-irts Fri May 2 09:17:12 1997 1

```
#!/usr/sbin/perl
#
# Usage: query-irts keyfile dirquery spec outfile
#
# first, make an array of the childids we are interested in, taken from file
# keydir (first column)
# then, run smart for each of them.
# store it in outfile.
# we should input a directory for the query files, and the spec file

# $ARGV[0]=file of child-parent pairs by message number
# $ARGV[1]=directory of single file queries, string ending with '/'; must contain
# /subjects/ , /quotes/ , or /unquotes/
# $ARGV[2]=location of corresponding spec file
# $ARGV[3]=where to put the output
$keydir=$ARGV[0];
$querydir=$ARGV[1];
$spec=$ARGV[2];
$outfile=$ARGV[3];


# first we build the array of the messageids we want to use as queries
#open(INDEX, "/radish/070/projtrain/child-parent-pairs-irt-id");
#open(INDEX, "/radish/070/projtest10k/child-parent-pairs-irt-id");
#open(INDEX, "child-parent-pairs-irt-id");
open(INDEX, $keydir);
while(<INDEX>) {
    $idnum=substr($_,0,5);
    push(@childids,$idnum);
}

if ($querydir=~/subjects/) {
    $suffix=".subj";}
elsif ($querydir=~/unquotes/) {
    $suffix=".uquot";}
elsif ($querydir=~/quotes/) {
    $suffix=".quot";}

$num_wanted=3000;

foreach (@childids) {
    $queryfile="$querydir$_$suffix";
    open(SMART, "runsmart $queryfile $spec $num_wanted|");
    open (OUTDIR, ">>$outfile");
    while (<SMART>) {
        if (/No documents to display/) {
            print OUTDIR "$_\n";
        }
        else {
            print OUTDIR;
        }
    }
    close (OUTDIR);
}
```

query-quoted Fri May 2 09:17:12 1997 1

```
#!/bin/ksh
#
# query-quoted PREFIX SPEC NUMOUT > OUT
#
# Description:
# Iterate over several documents (one
# per file), run them as smart queries, and save the
# results for the top 10 documents.
#
# history: adapted from smartbatch1
#
# set -x
prefix=$1       # directory of files which are indiv. messages
spec=$2         # location of spec file
numout=$3       # number of records per query

for file in $(prefix)*
do
  rm /tmp/tmp.query-quoted*
  echo $file
  cat $file | get-quoted > /tmp/tmp.query-quoted-data
  s=$(wc -c /tmp/tmp.query-quoted-data)
  s1=$(echo $s | nawk '{print $1}')
  if ((s1==0))
  then print "No quoted material"
  else
    cat ~/project/queryprefix /tmp/tmp.query-quoted-data /radish/070/tmp/querysuffix > /tmp/tmp.query-quoted-query
  cat /tmp/tmp.query-quoted-query ! smart inter $spec verbose 0 num_wanted $numout
  fi
done
exit 0
```

query-run Fri May 2 09:17:12 1997 1

```
#!/bin/ksh
#
# Usage: query-run PREFIX SPEC NUMOUT > OUT
#
# Description: This program is a demonstration of operating
# smart in batch mode.  Iterate over several documents (one
# per file), run them as smart queries, and save the
# results for the top 10 documents.
#
# This version runs on each file, in its entirety.  therefore, it is meant for use when the
# files are preparsed, and the entire file is used as a query.  This removes the need to
# filter out the query by piping it from another script.

# set -x
prefix=$1       # directory of files which are indiv. messages
spec=$2         # location of spec file
numout=$3       # number of records per query

for file in $(prefix)*
do

  rm /tmp/tmp.query-run*
  echo $file
  cat $file > /tmp/tmp.query-run-data
  s=$(wc -c /tmp/tmp.runsmart-data)
  s1=$(echo $s | nawk '{print $1}')
  if ((s1==0))
  then print "No quoted material"
  else
  cat /radish/070/tmp/queryprefix /tmp/tmp.query-run-data /radish/070/tmp/querysuffix > /tmp/tmp.query-run-query
  cat /tmp/tmp.query-run-query | smart inter $spec verbose 0 num_wanted $numout
  fi
done
exit 0
```

query-run-better Fri May 2 09:17:12 1997 1

```
#!/usr/sbin/perl
#
# Usage: query-run-better indir spec numout outfile
#
# similar to query-run: runs smart on each doc in indir across docs
# in spec.

$indir=$ARGV[0];
$spec=$ARGV[1];
$num_wanted=$ARGV[2];
$outfile=$ARGV[3];

opendir(INDIR, $indir);
@files=readdir(INDIR);
closedir(INDIR);
shift(@files);
shift(@files);

foreach (@files) {
    open(SMART, "runsmart $indir$_ $spec $num_wanted|");
    open (OUTDIR, ">>$outfile");
    while (<SMART>) {
        if (/No documents to display/) {
            print OUTDIR "$_\n";
        }
        else {
            print OUTDIR;
        }
    }
    close (OUTDIR);
}
```

runsmart Fri May 2 09:17:12 1997 1

```
#!/bin/ksh
#
# runsmart FILE SPEC NUMOUT
#
# Description: This program is a demonstration of operating
# smart in batch mode.  Iterate over several documents (one
# per file), run them as smart queries, and save the
# results for the top 10 documents.
#
#set -x
file=$1          # directory of files which are indiv. messages
spec=$2          # location of spec file
numout=$3        # number of records per query

rm /tmp/tmp.runsmart*
echo $file
cat $file > /tmp/tmp.runsmart-data
  s=$(wc -c /tmp/tmp.runsmart-data)
  s1=$(echo $s | nawk '{print $1}')
  if ((s1==0))
  then print "No quoted material"
  else
    cat /radish/070/tmp/queryprefix /tmp/tmp.runsmart-data /radish/070/tmp/querysuffix > /tmp/tmp.runsmart-query
    cat /tmp/tmp.runsmart-query | smart inter $spec verbose 0 num_wanted $numout
  fi


exit 0
```

```
David D. Lewis and Kimberly Knowles. 29-May-96

child-parent-rank-score: 8/22/95 scans through full-query-* files and prints
out the child:parent <rank> <score> for each combination of found retrieved
parent for the child
# usage: child-parent-rank-score INFILE OUTFILE
# Scans through full-query-* files and prints out the
# child.parent <rank> <score>
# for each retrieved parent for the child in OUTFILE.
#
# Assumes the lines in INFILE contain filenames of form "0nnnn." followed
# by suffixes quot, unquot, subj, or mbox.


-----------------------------------------
create_full_files: 8/25/95 a full "make" sort of file template.  meant to
operate from all stages of processing data, it is meant for copying to the
appropriate place and customizing location and style of data (mbox format
versus digest, what sort of lists of children and parents are used, how
much as to be processed, etc.)
# usage: create_full_files 2> cff-log
# shell program to run lots of stuff overnight:
#
# assumes a directory called messages/ in current dir, which contains
# indiv. messages that comprise the dataset.
# some global parameters.
homedir="/radish/070/projtestsibs"
dbase_name="projtestsibs"
indexfilename="child-sibling-pairs-irt-id"
message_source="/radish/070/projtest10k/messages/"
# if this is a source for sibling lists
pairs_list="/radish/070/projtest10k/child-parent-pairs-irt-id"
#be sure to check the make_unquoted_(digest-)files line.
  NOTE. THIS IS TOP LEVEL DO THE WHOLE THING


-----------------------------------------
finish-cprs: 8/11/95 to replace 'nr' in complete cprs file with appropriate
max numbers from num-retrieved-* files.
# usage: finish-cprs FILE > OUTFILE
#
# to run on an 11-column file of ranks, such as cprs-*.  will replace nr with
# 0.00 in the score column and the lowest rank from files that are hardwired:
@files=("num-retrieved-quotq-quot", "num-retrieved-quotq-unquot",
        "num-retrieved-unquotq-quot", "num-retrieved-unquotq-unquot",
        "num-retrieved-subj");
   NOTE. ASSIGNS RANK TO NONRETRIEVED ITEMS


-----------------------------------------
get-score: 7/18/95 takes a dataset file as arg, finds child (first) message
number, and parent (second) message number, opens a record of smart runs
using sub find_score, which finds the rank of 2nd when using first as a query,
based on data in a hardwired file, and prints results in 3 fields.
called on by make-score-list.
# usage: get-score FILE
#
# this is called on by script make-score-list, to get the message numbers of
# each child and parent, and the corresponding rank of the parent.

-----------------------------------------
kimjoin: 8/11/95 joins two cprs-* files together, and does the right thing.
# usage: kimjoin file1 file2 outfile
#
# takes two files, sorted numerically, of data of form
# <childid> <parentid>\t(<rank>\t<score>)x N
# and joins it the right way, with
# <childid>.<parentid>\t(<rank>\t<score>)x N1 \t (<rank>\t<score>)x N2
# replacing the right number of fields with nr when there was no data from
# the file.


-----------------------------------------
make-score: 8/4 adapted from make-score-list and get-score, takes a drun file
and created a score file.
# usage: make-score KEYFILE DRUN_FILE OUTFILE
#
# takes keyfile index of child-parent message numbers, file of output from running
# smart, and outfile, prints out childnum, parentnum, rank the parent came back at
# in outfile.  if parent not returned, says 'not in top $n', where $n is the number
# of documents returned.
# taken from ~/project/bin/ make-score-list and get-score, adapted for better
# I/O stuff.


-----------------------------------------
make-score-list: 7/18/95 prints labels in 3 fields, runs get-score for all
files in dir.
# make-score-list DIRECTORY > OUTFILE

-----------------------------------------
```

```
num-docs-retrieved: 8/10/95 counts up number of docs retrieved from full-query-* files
and prints them <child> <num_retrieved> in file.
# usage: num-docs-retrieved INFILE OUTFILE
# goes through a full-query-* file and prints to outfile the number of
# documents retrieved for each child.


-----------------------------------------
smartbatch1: 7/11/95 cats smart commands and results of mailbody1. and runs
smart on each file in dir.


-----------------------------------------
test-children: 7/25/95 like get-score and make-score-list, uses sub amichild to look
for two indicators that a message is a child, and runs it for every file in a prefix.
# Usage: test-children PREFIX OUTFILE NUMOUT
#
# model after make-score-list and get-score, except we want it all in perl, so
# we can recognize the filenames as well.
#
# okay, first we want to grab everything in a file.  i think we can pipe in
# filenames from shell command find to open (FILEHANDLE, "find <blah> |") or
# something.
# then we want to open each one and scan to see one of those things.
# then we want to print out what the final status was, childid and T or F
#
# we can use return $variable for passing values to and from subroutines.


-----------------------------------------
test-irt-better: 8/4/95 does the same as test-irt, only in 2n instead of n^2!
:)
# Usage: test-irt-better PREFIX OUTIDS
#
# meant to create a file for outputting the childid's and ppid's of
# messages which have message-id's in the in-reply-to header.
```

child-parent-rank-score Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# usage: child-parent-rank-score INFILE OUTFILE
# Scans through full-query-* files and prints out the
# child.parent <rank> <score>
# for each retrieved parent for the child in OUTFILE.
#
# Assumes the lines in INFILE contain filenames of form "0nnnn." followed
# by suffixes quot, unquot, subj, or mbox.

$infile=$ARGV[0];
$outfile=$ARGV[1];
open (FULL_QUERY, $infile);
open (OUTFILE, ">$outfile");
DATA:
while(<FULL_QUERY>) {
    if (/\/0+(\d+)\.[quot|unquot|subj|mbox]/) {
        $child=$1;
    }
    elsif (/^Num/) {
        $rank=0;
    }
    elsif (/^No [q|d]/) {
    }
    else {
        $rank++;
        ($parent,$score,$title)=/^\s+(\d+)\s+([01]\.\d+)\s( )/;
        print OUTFILE "$child.$parent\t$rank\t$score\n";
#       push(@data,"$child:$parent:$rank:$score");
    }
}
close(OUTFILE);
close(FULL_QUERY);
```

create_full_files Fri May 2 09:20:35 1997 1

```
#!/bin/ksh
#
# usage: create_full_files 2> eff-log
# shell program to run lots of stuff overnight.
#
# assumes a directory called messages. in current dir. which contains
# indiv. messages that comprise the dataset.

# some global parameters
homedir="/radish/070/projtestsibs"
dbase_name="projtestsibs"
indexfilename="child-sibling-pairs-irt-id"
message_source="/radish/070/projtest10k/messages/"
# if this is a source for sibling lists
pairs_list="/radish/070/projtest10k/child-parent-pairs-irt-id"
#be sure to check the make_unquoted_[digest-]files line.

mkdir $(homedir)
mkdir $(homedir)/messages
cd $(message_source)
forall "cp REPL $(homedir)/messages."
cd $(homedir)

# make and index of the answers
#test-irt-better messages/ child-parent-pairs-man-id
# or, for creating the sibling list.
cp $(pairs_list) $(homedir)/child-parent-pairs-irt-id
sort -n +1 child-parent-pairs-irt-id > sorted-cp-pairs
gec-sibs sorted-cp-pairs cs-pairs
sort -n cs-pairs > child-sibling-pairs-irt-id

# generate the data files on extracted text
make-quoted-files messages/ quotes.
rm quotes/.quot
#make-unquoted-files messages/ unquotes/
make-unquoted-digest-files messages. unquotes/
rm unquotes/.uquot
make-subj-files messages/ subjects/
rm subjects/.subj
date
print -u2 "extracted files made\n"

# build the smart databases
~/project/make_files/make_text_quot quotes/ /radish/070/kim-email-dbases/$(dbase_name)quot
~/project/make_files/make_text_uquot unquotes/ /radish/070/kim-email-dbases/$(dbase_name)unquot
~/project/make_files/make_text_subj subjects/ /radish/070/kim-email-dbases/$(dbase_name)subj
date
print -u2 "smart databases made\n"

# run queries, generate data
cd $(homedir)
query-irts $indexfilename quotes/ /radish/070/kim-email-dbases/$(dbase_name)quot/spec full-query-quotq-quot
date
print -u2 "full query qq made\n"
make-score $indexfilename full-query-quotq-quot full-parent-rank-quotq-quot
recall-precision-table full-parent-rank-quotq-quot rp-table-quotq-quot
child-parent-rank-score full-query-quotq-quot cprs-quotq-quot
num-docs-retrieved full-query-quotq-quot num-retrieved-quotq-quot
date
print -u2 "qq files completed\n"

query-irts $indexfilename quotes/ /radish/070/kim-email-dbases/$(dbase_name)unquot/spec full-query-quotq-unquot
date
print -u2 "full query qu made\n"
make-score $indexfilename full-query-quotq-unquot full-parent-rank-quotq-unquot
recall-precision-table full-parent-rank-quotq-unquot rp-table-quotq-unquot
child-parent-rank-score full-query-quotq-unquot cprs-quotq-unquot
num-docs-retrieved full-query-quotq-unquot num-retrieved-quotq-unquot
date
print -u2 "qu files completed\n"

query-irts $indexfilename unquotes/ /radish/070/kim-email-dbases/$(dbase_name)quot/spec full-query-unquotq-quot
date
print -u2 "full query uq made\n"
make-score $indexfilename full-query-unquotq-quot full-parent-rank-unquotq-quot
recall-precision-table full-parent-rank-unquotq-quot rp-table-unquotq-quot
child-parent-rank-score full-query-unquotq-quot cprs-unquotq-quot
num-docs-retrieved full-query-unquotq-quot num-retrieved-unquotq-quot
date
print -u2 "uq files completed\n"

query-irts $indexfilename unquotes/ /radish/070/kim-email-dbases/$(dbase_name)unquot/spec full-query-unquotq-unquot
date
print -u2 "full query u made\n"
make-score $indexfilename full-query-unquotq-unquot full-parent-rank-unquotq-unquot
recall-precision-table full-parent-rank-unquotq-unquot rp-table-unquotq-unquot
child-parent-rank-score full-query-unquotq-unquot cprs-unquotq-unquot
num-docs-retrieved full-query-unquotq-unquot num-retrieved-unquotq-unquot
date
print -u2 "uu files completed\n"

query-irts $indexfilename subjects/ /radish/070/kim-email-dbases/$(dbase_name)subj/spec full-query-subj
```

```
date
print -u2 "full query s made\n"
make-score $indexfilename full-query-subj full-parent-rank-subj
recall-precision-table full-parent-rank-subj rp-table-subj
child-parent-rank-score full-query-subj cprs-subj
num-docs-retrieved full-query-subj num-retrieved-subj
date
print -u2 "s files completed\n"

rm cprs.tmp
sort -n cprs-quotq-quot > cprs.tmp
rm cprs-quotq-quot
mv cprs.tmp cprs-quotq-quot
sort -n cprs-quotq-unquot > cprs.tmp
rm cprs-quotq-unquot
mv cprs.tmp cprs-quotq-unquot
sort -n cprs-unquotq-quot > cprs.tmp
rm cprs-unquotq-quot
mv cprs.tmp cprs-unquotq-quot
sort -n cprs-unquotq-unquot > cprs.tmp
rm cprs-unquotq-unquot
mv cprs.tmp cprs-unquotq-unquot
sort -n cprs-subj > cprs.tmp
rm cprs-subj
mv cprs.tmp cprs-subj
date
print -u2 "cprs files sorted\n"

kimjoin cprs-quotq-quot cprs-quotq-unquot cprs-qq-qu
date
print -u2 "cprs files concatenated qq-qu\n"
kimjoin cprs-qq-qu cprs-unquotq-quot cprs-qq-qu-uq
date
print -u2 "cprs files concatenated qq-qu-uq\n"
kimjoin cprs-qq-qu-uq cprs-unquotq-unquot cprs-qq-qu-uq-uu
date
print -u2 "cprs files concatenated qq-qu-uq-uu\n"
kimjoin cprs-qq-qu-uq-uu cprs-subj cprs-qq-qu-uq-uu-s
date
print -u2 "cprs files concatenated qq-qu-uq-uu-s\n"

finish-cprs cprs-qq-qu-uq-uu-s cprs-final
date
print -u2 "cprs files completed\n"

# now just a little cleanup
mkdir cprs-concat
mv cprs-qq* cprs-concat
mkdir cprs-origs
mv cprs-q* cprs-u* cprs-s* cprs-origs
mkdir full-queries
mv full-query-* full-queries
mkdir num-rets
mv num-retrieved* num-rets

exit 0
```

finish-cprs Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# usage: finish-cprs FILE OUTFILE
#
# to run on an 11-column file of ranks, such as cprs-*.  will replace nr with
# 0.00 in the score column and the lowest rank from files that are hardwired.

#@files=$ARGV[0];
@files=("num-retrieved-quotq-quot", "num-retrieved-quotq-unquot",
        "num-retrieved-unquotq-quot", "num-retrieved-unquotq-unquot",
        "num-retrieved-subj");

foreach (@files) {
    if ($_=~/\-quotq\-quot/) {
        $file="1";
    }
    elsif ($_=~/\-quotq\-unquot/) {
        $file="3";
    }
    elsif ($_=~/\-unquotq\-quot/) {
        $file="5";
    }
    elsif ($_=~/\-unquotq\-unquot/) {
        $file="7";
    }
    elsif ($_=~/\-subj/) {
        $file="9";
    }

    open(MAXFILE, $_);
    while (<MAXFILE>) {
        ($child, $maxval)=split(/\s/,$_,2);
        chop($maxval);
#       print "$child\n$maxval\n";
#       @dataarray{join(".",$file,$child)}=$maxval;
        $dataarray{join(".",$file,$child)}=$maxval;
    }
    close(MAXFILE);
}

#foreach $stuff (sort keys(%dataarray)) {
#    $ans=$dataarray{$stuff};
#    print "$stuff\t$ans\n";
#}

#@ref= (1,3,5,7,9);
open(CPRS, $ARGV[0]) || die "Couldn't open infile";
open(OUTFILE,">$ARGV[1]") || die "Couldn't open outfile";
while (<CPRS>) {
    chop;
    s/nr/0\.00/g;
#    print "$_\n";
#    ($childparent,$r1,$s1,$r2,$s2,$r3,$s3,$r4,$s4,$r5,$s5)=split(/\s/,$_,11);
    @currentline=split(/\s/,$_,11);
#    print "$childparent,$r1,$s1,$r2,$s2,$r3,$s3,$r4,$s4,$r5,$s5\n";
    ($childid,$parentid)=split(/\./,@currentline[0],2);
#    print "Child: $childid\tParentid: $parentid\n";
    foreach $ref (1,3,5,7,9) {
#       print "currentline[$ref]=@currentline[$ref]\n";
        if (@currentline[$ref]=="0.00") {
            @currentline[$ref]="$dataarray{join(".",$ref,$childid)}\n";
#           print "Replacing column $ref of $_ with $dataarray{join(".",$ref,$childid)}\n";
        }
    }
    print OUTFILE join("\t",@currentline),"\n";
#    print "$childparent,$r1,$s1,$r2,$s2,$r3,$s3,$r4,$s4,$r5,$s5\n";
}
close(OUTFILE);
```

get-score Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# usage: get-score FILE
#
# this is called on by script make-score-list, to get the message numbers of
# each child and parent, and the corresponding rank of the parent.
#

$state=0;

while (<>) {
    if ($state==0) {
        if (/Message #(\d+):/) {
            $child=$1;
            $state=1;
        }
    }
    elsif ($state==1) {
        if (/Message #(\d+):/) {
            $parent=$1;
            &print_results;
        }
    }
}

sub print_results {
    $score=&find_score($child,$parent);
    if ($score==nil) {
        $score="null score";
    }
    elsif ($score>10) {
        $score="not top 10";
    }
    print "$child\t$parent\t$score\n";
    $state=0;
}

sub find_score {
#    open(INFILE, "/home/kknowles/project/stats/wwwtalk/drun-children") || die "Can't find file";
    open(INFILE, "/home/kknowles/project/stats/wwwtalk/query-by-quote-long") || die "Can't find file";
    $state=0;
    while (<INFILE>) {
#        print "$_\n$rank\n";
        if ($state==0) {
            if (/0+$child/) {
                $state=1;
                $rank=0;
            }
        }
        elsif ($state==1) {
            if (/^\s+$parent\s/) {
                ($s=/^\s+$parent\s+//);
                return $rank;
            }
            elsif (/^No quoted material/) {
                $rank="No quoted material.";
                return $rank;
            }
            elsif (/^No documents to display/) {
                $rank="No documents to display.";
                return $rank;
            }
            $rank++;
        }
        else {
            die "Shouldn't get here";
        }
    }
}
```

kimjoin Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# usage: kimjoin file1 file2 outfile
#
# takes two files, sorted numerically, of data of form
# <childid>.<parentid>\t(<rank>\t<score>)x N
# and joins it the right way, with
# <childid>.<parentid>\t(<rank>\t<score>)x N1 \t (<rank>\t<score>)x N2
# replacing the right number of fields with nr when there was no data from
# the file.

open(OUTFILE, ">$ARGV[2]");

open(FILEA, $ARGV[0]);
open(FILEB, $ARGV[1]);
$a=<FILEA>;
$b=<FILEB>;
chop($a);
chop($b);
$anr=$a;
($at,$anr)=split(/\s/,$a,2);
$anr=~s/[^\t]+/nr/g;
#$anr=~s/[^\t]+/0\.00/g;
$bnr=$b;
($bt,$bnr)=split(/\s/,$b,2);
$bnr=~s/[^\t]+/nr/g;
#$bnr=~s/[^\t]+/0\.00/g;
#print "$anr\t$bnr\n";

while (1) {
    if ($a=="" && $b=="") {
        exit;
    }
    ($a1,$a2)=split(/\s/,$a,2);
    ($b1,$b2)=split(/\s/,$b,2);
#     print "a1=$a1\tb1=$b1\n";
#     print "a2=$a2\tb2=$b2\n";
    if ($a1 == $b1) {
        if ($a1 eq $b1) {
            print OUTFILE "$a1\t$a2\t$b2\n";
            $a=<FILEA>;
            $b=<FILEB>;
            chop($a);
            chop($b);
        }
        elsif (length($a1)<length($b1)) {
            print OUTFILE "$a1\t$a2\t$bnr\n";
            $a=<FILEA>;
            chop($a);
        }
        elsif (length($a1)>length($b1)) {
            print OUTFILE "$b1\t$anr\t$b2\n";
            $b=<FILEB>;
            chop($b);
        }
    }
    elsif (((0<$a1) && ($a1 < $b1)) || ($b1=="")) {
        print OUTFILE "$a1\t$a2\t$bnr\n";
        $a=<FILEA>;
        chop($a);
    }
    elsif (((($a1 > $b1)) && (0<$b1)) || ($a1=="")) {
        print OUTFILE "$b1\t$anr\t$b2\n";
        $b=<FILEB>;
        chop($b);
    }
    else {
        die "shouldn't get here";
    }
}

close(OUTFILE);
```

make-score Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# usage: make-score KEYFILE DRUN_FILE OUTFILE
#
# takes keyfile index of child-parent message numbers, file of output from running
# smart, and outfile, prints out childnum, parentnum, rank the parent came back at
# in outfile.  if parent not returned, says 'not in top Sn', where Sn is the number
# of documents returned.
# taken from ~/project/bin/ make-score-list and get-score, adapted for better
# I/O stuff.
$keyfile=$ARGV[0];              # file which contains desired child-parent pair by msgnum
$drunfile=$ARGV[1];             # file which is the output of running smart
$outfile=$ARGV[2];              # output file

# this to form $child $parent data array
open(INFILE, $keyfile),
    while (<INFILE>) {
        chop;
        push(@pairs,$_);
    }
close(INFILE);


open(OUTFILE, ">$outfile") || die "Couldn't open outfile".
open(INFILE, $drunfile) || die "Couldn't open infile";

$rank="Rank";
#print OUTFILE "ChildID\tPParID\tRank\n";

LOOPARRAY:
foreach (@pairs) {
    print OUTFILE "$child\t$qparent\t$rank\n" unless ($rank eq "Rank");
    &new_pairs;
    if ($tmp=~/$child\.[quot|uquot|subj]/) {
        $state=1;
        $rank=0;
    }
    while(<INFILE>) {
        if ($state==0) {
            if (/$child\.[quot|uquot|subj]/) {
                $state=1;
                $rank=0;
            }
        }
        elsif ($state==1) {
            if (/^\s+$parent\s/) {
                (s/^\s+$parent\s+//);
                next LOOPARRAY;
            }
            elsif (/^No quoted material/) {
                $rank="No quoted material.";
                next LOOPARRAY;
            }
            elsif (/^No documents to display/) {
                $rank="No documents to display.";
                next LOOPARRAY;
            }
            elsif (/0\d\d\d\d\.[quot|uquot|subj]/) {
                $rank=$rank-1;  # correction for the line itself being counted
                $rank="Not in top $rank.";
                $tmp=$_;
                next LOOPARRAY;
            }
            $rank++;
        }
        else {
            die "Shouldn't get here";
        }
    }
}

print OUTFILE "$child\t$qparent\t$rank\n";
close(OUTFILE),

sub new_pairs {                 # assigns each pair to $child, $parent
#    ($child,$parent) = split(/\t\t/,$_,2);
    ($child,$parent) = split(/\t/,$_,2);
    $qparent=$parent;
    $parent=~s/^0+//;
#    print "Read new pair $child, $parent\t";
    $state=0,
}
```

make-score-list Fri May 2 09:20:35 1997 1

```
#!/bin/ksh
#
# make-score-list DIRECTORY > OUTFILE
#
prefix=$1       # directory of files which are indiv. datasets

#print ${prefix}
#print "Child\tParent\tRank\n"
for file in ${prefix}*
do
  get-score $file
done
exit 0
```

num-docs-retrieved Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# usage: num-docs-retrieved INFILE OUTFILE
# goes through a full-query-* file and prints to outfile the number of
# documents retrieved for each child.

open (FULL_QUERY, $ARGV[0]);
open (OUTFILE, ">$ARGV[1]");
while(<>) {
    if (/\/0+(\d+)\.(quot|unquot|subj|mbox)/ || eof) {
        print OUTFILE "$child\t$num\n" unless $child=="";
        $child=$1;
    }
    elsif (/^N/) {
        $num=0;
    }
    else {
        $num++;
    }
}
close(FULL_QUERY);
close(OUTFILE);
```

smartbatch1 Fri May 2 09:20:35 1997 1

```
#!/bin/ksh
#
# smartbatch1 PREFIX SPEC NUMOUT > OUT
#
#
# set -x
prefix=$1       # directory of files which are indiv. messages
spec=$2         # location of spec file
numout=$3       # number of records per query

for file in $(prefix)*
do
  rm /tmp/tmp.smartbatch1*
  echo $file
  cat $file | mailbody1 > /tmp/tmp.smartbatch1-data
  cat /radish/070/tmp/queryprefix /tmp/tmp.smartbatch1-data /radish/070/tmp/querysuffix > /tmp/tmp.smartbatch1-query
  cat /tmp/tmp.smartbatch1-query | smart inter $spec verbose 0 num_wanted $numout
done
exit 0
```

test-children Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# Usage: test-children PREFIX OUTFILE NUMOUT
#
# model after make-score-list and get-score, except we want it all in perl, so
# we can recognize the filenames as well.
#
# okay, first we want to grab everything in a file.  i think we can pipe in
# filenames from shell command find to open (FILEHANDLE, "find <blah> |") or
# something.
# then we want to open each one and scan to see one of those things.
# then we want to print out what the final status was, childid and T or F.
#
# we can use return $variable for passing values to and from subroutines.

#open(OUTFILE, ">>$ARGV[1]") ||die "Couldn't open outfile";
open(OUTFILE, ">$ARGV[1]") ||die "Couldn't open outfile";

print OUTFILE "Message ID\tChild?\n";

#open(FIND, "find $ARGV[0] -type f -name '*.mbox' -print|head -$ARGV[2]|") || die "Couldn't run find: $!\n";
open(FIND, "find $ARGV[0] -type f -name '*.mbox' -print|") || die "Couldn't run find: $!\n";
# the second one does it for all; the first for a specified number of messages.

while ($filename=<FIND>) {
$msgid=&get_child_id($filename);
$answer=&amichild($filename);
print OUTFILE "$msgid\t\t$answer\n";
}

# we can try to extract the actual child id from the filename, i guess.
sub get_child_id {
    $filename =~ /(\d+)\.mbox/;
    return $1;
}

# used to search for quoted material
$pat2 = "([A-Za-z]*>+)";
$pat3 = "(\\*)";
$pat4 = "([A-Za-z]*\\|+)";
$pat = "$pat2|$pat3|$pat4";

sub amichild {
# takes a file as an argument.  looks for In-Reply-To: header, Re: in Subject
# header, and quoted material.  If all are absent, returns false.  else,
# returns true.
    open(MSGFILE, $filename) || die "Can't open file";
    while (<MSGFILE>) {
        if (/^In-Reply-To:/) {
            return "T";
        }
        elsif (/^Subject: Re:/) {
            return "T";
        }
# we don't seem to need to scan for quoted material.  i think the subject line
# containing the Re: is a really robust example.  we'll leave it for now.
#       elsif ( /^> /) {
#       if (/^>  ) {
#           return "T";
#       }
#       elsif ( /^\s*)($pat)($pat|(\s))*/) {
#           return "T";
#       }
    }
    return "F";
}
```

```
#!/usr/sbin/perl
#
# Usage: test-irt PREFIX OUTIDS OUTNAME [NUMOUT]
#
# meant to create two files: one for outputting the childid's and ppid's of
# messages which have message-id's in the in-reply-to header, and one containing
# the childid's and ppid's of the messages which have author name and date/time
# in the irt header.
#
# okay, the way this will work is first we will open two outfiles.
# print the headers for each file.
# then we will open each infile.
# if it contains the irt field (optionally we can save the line to a variable)
# we want to see if it contains the pattern /(<[A-Za-z0-9\.]+@[A-Za-z0-9.]+>)/.
# if it does, we want to capture that $1 and look through all the other messages
# in the file until we find /Message-Id: $1/.  then we want to get the number of
# the current file.
# then print childid and ppid.
# then we need to close the child file and do the same for the next.

if ($ARGV[3]=="") {
    $ARGV[3]="3000";
}
open(OUTIDS, ">$ARGV[1]") ||die "Couldn't open outfile";
print OUTIDS "Message ID\tPParent Id\n";
open(OUTNAME, ">$ARGV[2]") ||die "Couldn't open outfile";
print OUTNAME "Message ID\tPParent Id\n";

#open(INFILE, "/bin/ls -1 $ARGV[0] |head -$ARGV[3]|") || die "Couldn't run ls: $!\n";
# this one (above) goes through in forwards order.
#open(INFILE, "find $ARGV[0] -type f -name '*.mbox' -print|head -$ARGV[3]|"); || die "Couldn't run find. $!\n"

opendir(INDIR,$ARGV[0]);
@infile=readdir(INDIR);
closedir(INDIR);

#while ($filename=<INFILE>) {
foreach (@infile) {
    $filename=$_;
#    print $filename;
    if ($state=&irt) {
        $msgid=&get_msg_id($filename);
        if ($state==1) {
            $answer=&find_parent_id($filename,$parentid);
            print OUTIDS "$msgid\t\t$answer\n";
        }
        elsif ($state==2) {
            $answer=&find_parent_name(filename);
            print OUTNAME "$msgid\t\t$answer\n";
        }
    }
}

# we can try to extract the actual id from the filename, i guess
sub get_msg_id {
    local($fname)=@_;
    $fname =~ /(\d+)\.mbox/;
    return $1;
}

sub irt {
    $/="";
    $*=1;
#    print $ARGV[0] . "/" . $filename;
    open(MSGFILE, $ARGV[0] . "/" . $filename) || die "Can't open file $filename:$!";
    while (<MSGFILE>) {
        if (/^In-Reply-To:[^<]*(<[A-Za-z0-9\.-]{9,100}@[A-Za-z0-9\.-]+>)/) {
            $parentid=$1;
#            print "Parent ID= $parentid";
            $/="\n";
            $*=0;
            close(MSGFILE);
            return "1";
        }
        elsif (/^In-Reply-To: (.*)/) {
            $irt_text=$1;
            $/="\n";
            $*=0;
            close(MSGFILE);
            return "2";
        }
    }
    $/="\n";
    $*=0;
    close(MSGFILE);
    return "0";
}

sub find_parent_id {
#    open(PAR, "/bin/ls -1 $ARGV[0] |head -$ARGV[3]|") || die "Couldn't run find: $!\n";
#    while ($current=<PAR>) {
    foreach (@infile) {
```

```
        $filename=$_;
        open(ENT, $ARGV[0] . $current) || die "Couldn't open parent file",
        while (<ENT>) {
            if (/^Message-Id: $parentid/) {
                close(ENT);
#               close(PAR);
                return &get_msg_id($current);
            }
        }
        close(ENT);
    }
#    close(PAR);
    return $parentid;
}

sub find_parent_name {
# okay, so now we need to do two things: collect appropriate info from the irt
# line in the child, and then match the parent that those things fit.
# i think we should get a list of parents that satisfy those conditions (the
# exact time being a stronger statement than the date) and find where the arrays
# intersect.  we may need only the time to match up.

#    &get_time_of_child;
#    &find_parents_with_time;
    return "does not contain id number";
}

$pat2="(\d{2,2}:\d{2,2}:\d{2,2})";

sub get_time_of_child {
    if ($irt_text =~ /($pat2)/) {
        $child_time=$1;
    }
}

sub find_parents_with_time {
    if ($child_time) {
#       open(PAR, "/bin/ls -1 $ARGV[0] |head -$ARGV[3]|") || die "Couldn't run find: $!\n";
#       open(PAR, "/bin/ls -1 $ARGV[0]|") || die "Couldn't run find: $!\n";
#       while ($current=<PAR>) {
        foreach (@infile) {
            $filename=$_;
            open(ENT, $ARGV[0] . $current) || die "Couldn't open parent file";
            while (<ENT>) {
                if (/^Date: .$child_time/) {
                    close(ENT);
#                   close(PAR);
                    return &get_msg_id($current);
                }
            }
            close(ENT);
        }
#       close(PAR);
        return $child_time;
    }
    else {
        return "no time provided";
    }
}

sub get_date_of_child {
}

sub find_parents_with_date {
#    $pat1="([A-Z][a-z\.]+ [A-Z ]?[a-z\.]* [A-Z][a-z]+)";
#    $pat3="(";
#    open(PAR, "/bin/ls -1 $ARGV[0] |head -$ARGV[3]|") || die "Couldn't run find: $!\n";
#    while ($current=<PAR>) {
#    open(ENT, $ARGV[0] . $current) || die "Couldn't open parent file";
#    @months = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec');
#       while (<ENT>) {
#            foreach $i (0 .. $#months) {
#                if (/Date: $months[$i]/) {
#                $i++;
#            }
#
#       }
#    }
#    close(PAR);
#    return $parentid;
}
```

test-irt-better Fri May 2 09:20:35 1997 1

```
#!/usr/sbin/perl
#
# Usage: test-irt-better PREFIX OUTIDS
#
# meant to create a file for outputting the childid's and ppid's of
# messages which have message-id's in the in-reply-to header.
#

#if ($ARGV[2]=="") {
#    $ARGV[2]="3000";
#}

opendir(INDIR,$ARGV[0]);
@infile=readdir(INDIR);
closedir(INDIR);

#this section to make the array of fileid's and Message-Id's
foreach $filename (@infile) {
#    print "$filename\n";
    open(MSGFILE, $ARGV[0] . $filename) || die "Can't open file $filename:$!";
    $filename=~/(0\d+)\.mbox/;
    $parent=$1;
    while (<MSGFILE>) {
        if (/^Message-Id: (<[A-Za-z0-9\.-]{9,100}@[A-Za-z0-9\.-]+>)/) {
            $msgid=$1;
            $data="$parent\t$msgid";
#           print "$parent\t$msgid\n";
            push(@messageids,$data);
        }
    }
    close (<MSGFILE>);
}

#print "@infile";
#this section to make the array of fileid's and Message-Id's
foreach $filename (@infile) {
#    print "$_\n";
    open(MSGFILE, $ARGV[0] . $filename) || die "Can't open file $filename:$!";
    $filename=~/(0\d+)\.mbox/;
    $child=$1;
#    print "$child\n";
    while (<MSGFILE>) {
        if (/^In-Reply-To:[^<]*(<[A-Za-z0-9\.-]{9,100}@[A-Za-z0-9\.-]+>)/) {
            $parentid=$1;
            $data="$child\t$parentid";
#           print "$child\t$parentid\n";
            push(@irtids,$data);
        }
    }
    close (<MSGFILE>);
}

#now we try to match the parentids of @irtids with the msgids of @messageids
foreach (@irtids) {
    ($child,$parentid) = split(/\t/,$_,2);
    foreach (@messageids) {
        ($parent,$msgid) = split(/\t/,$_,2);
        if ($msgid eq $parentid) {
            $pair="$child\t\t$parent";
            push(@answers,$pair);
        }
    }
}

#now print out array in outfile.
open(OUTIDS, ">$ARGV[1]") ||die "Couldn't open outfile";
#print OUTIDS "Message ID\tPParent Id\n";
foreach (@answers) {
    print OUTIDS "$_\n";
}
```

README2-utils Fri May 2 09:20:23 1997 1

David D. Lewis and Kimberly Knowles, 29-May-96

---

forall: 8/17/95 does something for all files in current directory. either does the command for each file, or replaces each file for 'R' in command. (use R instead of *)
usage: forall "COMMAND PARAM"
forall "COMMAND REPL REPL'"

---

geomean: 8/14/95 calculates the geometric mean for set of data in array, of form cprs-final.
usage: geomean FILE > OUTFILE
#
to run on an 11-column file of ranks, such as cprs-final. will calculate
geometric mean for each child and parent pair.

---

rand-selector: 7/10/95 what you think it does. takes args maxnum and numwanted and gives back a list of random numbers. run a one-liner from the camel book to cut out the repetitions.

---

stdevchildparent: 7/17/95 finds ave and standard dev of a hardwired list of numbers, and prints it.

forall Fri May 2 09:20:23 1997 1

```
#!/usr/sbin/perl
#
# usage: forall "COMMAND PARAM"
#        forall "COMMAND REPL REPL"
#

opendir(INDIR,".");
    @allfiles=readdir(INDIR);
closedir(INDIR);

# cheap hack to eliminate ./ and ../ entries
shift(@allfiles);
shift(@allfiles);

foreach (@allfiles) {
    if ($ARGV[0]=~/REPL/) {
        $ARGV[0]=~s/REPL/$_/g;
        print "$ARGV[0]";
        $ARGV[0]=~s/$_/REPL/g;
    }
    else {
        print "$ARGV[0] $_";
    }
}
```

geomean Fri May 2 09:20:23 1997 1

```
#!/usr/sbin/perl
#
# usage: geomean FILE > OUTFILE
#
# to run on an 11-column file of ranks, such as cprs-final.  will calculate
# geometric mean for each child and parent pair.

open(CPRS, $ARGV[0]);
while (<CPRS>) {
    $prod=1;
    chop;
    s/nr/0\.00/g;
    @currentline=split(/\s/,$_,11);
    ($childid,$parentid)=split(/\./,@currentline[0],2);
    foreach $ref (1,3,5,7,9) {
        if (@currentline[$ref]=~/n/) {
            $tmp=@currentline[$ref];
            $tmp=~s/n//;
            $tmp=($tmp+1)/2;
        }
        else {
            $tmp=@currentline[$ref];
        }
        $prod=$prod*$tmp;
    }
    $geomean=$prod**.2;
    print "@currentline[0]\t$geomean\n";
}
```

rand-selector Fri May 2 09:20:23 1997 1

```
#!/usr/sbin/perl
#
# usage: rand-selector NUMWANTED MAXNUM > OUTFILE
# example: rand-selector 5 10 > tmp
#

srand(time|$$);
$i=@ARGV[0];
    while ($i>0) {

$a=int(rand(@ARGV[1]-1))+1;
print "$a\n";
$i--;}
```

stdevchildparent Fri May 2 09:20:23 1997 1

```
#!/usr/sbin/perl
#
# usage: stdevchildparent
#
# this should compute the standard dev of a list given it
#
#@data=(2,4,2,7,7,2,2,2,10,2,4,2,2,3,2,3,2,2,2,6,2,2,2,4,2,5);
#@data=(2,4,2,7,7,2,2,2,10,2,4,2,2,3,2,3,2,2,2,6,2,2,2,4,2,5,92);
#@data=(2,2,2,4,4,2,2,2,2,3,3,2,8,3,10,3,2,7,2,2,2,2,3,2,5,2,2,5,2,2,2,3,3,2,2,3,3,
        2,2,2,3,4,4,2,10,2,2,4,5,6,2,2,6,7,2,3,5,3,2,2,2,2);
@data=(2,2,2,4,4,2,3,2,3,172,3,26,2,8,3,10,3,437,2,7,2,2,12,116,2,2,3,2,5,2,2,13,
        5,2,2,2,3,3,2,2,3,3,2,2,2,3,4,26,4,2,10,2,2,4,5,6,2,1
,8,2,69,6,7,2,3,5,3,2,22,2,2,2,288);
#@data=(1,2,3);

$len=@data;
#print $len;
$i=$len-1;
$sum=0;
while ($i>-1) {
    $sum=$sum+$data[$i];
    $i--;
}
$ave=$sum/$len;
$i++;
$var=0;
while ($i<$len) {
    $d=$data[$i]-$ave;
#    print "$d\n";
    $var=$var+($d*$d);
    $i++;
}
$var=$var/($len-1);
$stdev=sqrt($var);
print "Average = $ave\n";
print "Standard Deviation = $stdev\n";
```

004-notes-using-smart-to-find-par Fri May 2 09:30:35 1997 1

```
(28-May-96 : Additional notes on how SMART was used to match portions
of child messages with potential parents  These notes supplement
the description in

@article(LEWIS96e
. author       = ''David D. Lewis and Kimberly A. Knowles''
. title        = ''Threading Electronic Mail. A Preliminary Study''
. journal      = ''Submitted for publication. Comments welcome.''
. year         = 1996
. copyrightown = ''AT&T 1996''
)

)

7/6/95: This is an early study of the messages that were picked out by smart
on a randomly chosen (I picked a number out of my head) article.  This used
the connectionists database, in /radish/070/netlists/archives/connectionists/

observation: the numbers that smart assigns seems to be (at least in the early
articles) just an index of the articles in the order it took them.  this is
true at least within the first archive file.  I haven't broken up the messages
into individual files yet.

I selected #23 to use as a query.  (remember that the query commands are case-
sensitive.)  The subject line is 'Kolmogorov's superposition theorem'.

Here are the smart results.

Smart (ntq?). Drun 23

-------------------------
Drun 23
----
Num  Action Sim Title
    23      1.00 Kolmogorov's superposition theorem
    43      0.28
    26      0.27 Kolmogorov's superposition theorem
   246      0.24 papers on pattern recognition
    59      0.23 NIPS CALL FOR PAPERS
   303      0.23 Bi-monthly Reminder file in Inbox. Composite networks TR: 'Conn
    20      0.23 CALL FOR PARTICIPATION
   321      0.23 Bi-monthly Reminder file in Inbox. Delays in Neuroprose Putting
   124      0.22 What is a connectionist net? Here's what it's not.
   221      0.22 NN conference. ROOMMATES FOR NIPS 89? CFP. Parallel Computing s

I will include the full text of the top three messages below these
observations.

#23 header info:
Date: Tue, 17 Jan 89 14:08:03 EST
From: sontag@fermat.rutgers.edu
Message-Id: <8901171908.AA00964@control.rutgers.edu>
To: Connectionists@cs.cmu.edu
Cc: rui@zeta.rice.edu
Subject: Kolmogorov's superposition theorem
Reply-To: sontag@fermat.rutgers.edu

#43 header info:
Date: Wed, 25 Jan 89 17:34:38 CST
From: Rui DeFigueiredo <rui@rice.edu>
Message-Id: <8901252334.AA01804@zeta.rice.edu>
To: Connectionists@cs.cmu.edu
Cc: bradley@fermat.rutgers.edu, poggio@wheaties.ai.mit.edu,
        sontag@fermat.rutgers.edu

        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

        In-Reply-To:  poggio@wheaties.ai.mit.edu's message of Tue, 17 Jan 89 22:
47:17 EST
        Subject  Kolmogorov's superposition theorem

# 26 header info:
Date: Tue, 17 Jan 89 22:47:17 EST
From: poggio@wheaties.ai.mit.edu (Tomaso Poggio)
Message-Id: <8901180347.AA21088@rice.chex.ai.mit.edu>
To: sontag@fermat.rutgers.edu
Cc: Connectionists@cs.cmu.edu, rui@zeta.rice.edu
In-Reply-To: sontag@fermat.rutgers.edu's message of Tue, 17 Jan 89 14:08:03 EST <8901171908.AA00964@control.rutgers.edu>
Subject: Kolmogorov's superposition theorem

Preliminary observations show that smart did very well with these particular
messages.  I however, have not searched to see if there are other messages it
should have gotten, though it seems implausible, as I have egreped the entire
archive for "Kolmogorov" (inserted below).  The tree of parent/child
relationships is as follows:
23>26>43.
Thus, smart got that they are interrelated, but ranked the
grandchild above the child.  However, the appropriate information can be gotten
by using just the message id.  The Subject: header is reliable only in content;
no Re: flags were inserted at all.  The dates alone could have shown the
```

correct order.

observation: in investigating whether or not smart was able to build the entire archive before radish crashed, I came upon the following problem. There is indeed a message that contains "^From my point of view as a connectionist," which is mistakenly parsed as a new message. Furthermore, in that same archive, arch.9106, the messages following that false new message are missing the pinhead headers. Thus, smart thinks it's one big message for who knows how many more messages. I doubt that the actual case is that this one message actually encapsulated so many forwards; whether it's an error in the archive or the smart parser, I don't know, but clearly perhaps the pinhead header is not as reliable as we once thought.

okay, i've found it out. smart did get to finish, as the last message in the archive is in the smart database. however, it is the nth message that got concatenated together (without the pinhead to separate them) in message 382.

---
titles

Num Action Sim Title
382 * 0.24 ph.d. thesis available: NN for Signal Processing NNSP95 - Forma
329 0.19 TD-Gammon paper available in neuroprose Brisbane Neural Network
278 0.16 Informal Computing Workshop Announcement: Conference for Philos
105 0.16 network meeting announcement for distribution
162 0.16 network meeting announcement
305 0.15 Why does the error rise in a SRN? Special Issue on Neural Model
351 0.14 Copernicus project with Central and Eastern European Countries
324 0.14 policy on posting talk announcements European Society for Philo
174 0.14 Share hotel room at IJCNN?
364 0.14 de Garis PerAc Report Some activity announc.--- molec. sys/engr

i just used the last screen or so of the last message as a query. the rest is history....

Dtext 23

From Connectionists-Request@Q.CS.CMU.EDU Tue Jan 17 14:43:00 1989
Received: from Q.CS.CMU.EDU by B.GP.CS.CMU.EDU; 17 Jan 89 14:42:24 EST
Received: from CS.CMU.EDU by Q.CS.CMU.EDU; 17 Jan 89 14:38:24 EST
Received: from FERMAT.RUTGERS.EDU by CS.CMU.EDU; 17 Jan 89 14:37:28 EST
Received: by control.rutgers.edu (5.59/(RU-Router/1.1)/3.01)
id AA00964; Tue, 17 Jan 89 14:08:03 EST
Date: Tue, 17 Jan 89 14:08:03 EST
From: sontag@fermat.rutgers.edu
Message-Id: <8901171908.AA00964@control.rutgers.edu>
To: Connectionists@cs.cmu.edu
Cc: rui@zeta.rice.edu
Subject: Kolmogorov's superposition theorem
Reply-To: sontag@fermat.rutgers.edu

*** I am posting this for Professor Rui de Figuereido, a researcher in Control Theory and Circuits who does not subscribe to this list. Please direct cc's of all responses to his e-mail address (see below)
-eduardo s. ***

KOLMOGOROV'S SUPERPOSITION THEOREM AND ARTIFICIAL NEURAL NETWORKS

Rui J. P. de Figueiredo
Dept. of Electrical and Computer Engineering
Rice University, Houston, TX 77251-1892
e-mail: rui@zeta.rice.edu

The implementation of the Kolmogorov-Arnold-Sprecher Superposition Theorem [1-3] in terms of artificial neural networks was first presented and fully discussed by me in 1980 [4]. I also discussed, then [4], applications of these structures to statistical pattern recognition and image and multi-dimensional signal processing. However, I did not use the words "neural networks" in defining the underlying networks. For this reason, the current researchers on neural nets including Robert Hecht-Nielsen [5] do not seem to be aware of my contribution [4]. I hope that this note will help correct history.

Incidentally, there is a misprint in [4]. In [4], please insert "no" in the statement before eqn.(4). That statement should read: "Sprecher showed that lambda can be any nonzero number which satisfies no equation ..."

[1] A.K.Kolmogorov, "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition," Dokl.Akad.Nauk.SSSR,Vol.114,pp.369-373,1957.
[2] V.I.Arnol'd, "On functions of three variables," Dokl.Akad.Nauk.SSSR, Vol.114,pp.953-956,1957.
[3] D.A.Sprecher, "An improvement in the superposition theorem of Kolmogorov," J.Math.Anal.Appl.,Vol.38,pp.208-213,1972.
[4] Rui J.P.de Figueiredo, "Implications and applications of Kolmogorov's superposition theorem,"IEEE Trans.Auto.Contr.,Vol.AC-25,pp.1227-1231,1980.
[5] R.Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem," IEEE 1st Int.Conf.on Neural Networks, San Diego,CA,June 21-24,1987,paper III-11.

Dtext 43

From Connectionists-Request@Q.CS.CMU.EDU Wed Jan 25 18:53:58 1989
Received: from Q.CS.CMU.EDU by B.GP.CS.CMU.EDU; 25 Jan 89 18:52:40 EST
Received: from CS.CMU.EDU by Q.CS.CMU.EDU; 25 Jan 89 18:47:04 EST
Received: from RICE.EDU by CS.CMU.EDU; 25 Jan 89 18:45:19 EST
Received: from zeta.rice.edu by rice.edu (AA29505); Wed, 25 Jan 89 17:44:04 CST
Received: by zeta.rice.edu (AA01804); Wed, 25 Jan 89 17:34:38 CST
Date: Wed, 25 Jan 89 17:34:38 CST
From: Rui DeFigueiredo <rui@rice.edu>
Message-Id: <8901252334.AA01804@zeta.rice.edu>
To: Connectionists@cs.cmu.edu
Cc: bradley@fermat.rutgers.edu, poggio@wheaties.ai.mit.edu,
sontag@fermat.rutgers.edu

- -

In-Reply-To: poggio@wheaties.ai.mit.edu's message of Tue, 17 Jan 89 22:47:17 EST
Subject: Kolmogorov's superposition theorem

Kolmogorov's theorem and its relation to networks are discussed in Biol. Cyber., 37, 167-186, 1979. (On the representation of multi-input systems: computational properties of polynomial algorithms. Poggio and Reichardt). There are references there to older papers (see especially the two nice papers by H. Abelson).

- - - - - - - - - - - - - - end of message - - - - - - - - - - - - -

Comment:

Poggio and Reichardt's paper, "On the representation of multi-input systems: Computational properties of polynomial algorithms" (Biol. Cyber., 37, 167-186, 1980) appeared, not earlier but, in the same year as deFigueiredo's, "Implications and applications of Kolmogorov's superposition theorem" (IEEE Trans. on Automatic Control, AC-25, 1227-1231, 1980).

Dtext 26

From Connectionists-Request@Q.CS.CMU.EDU Wed Jan 18 13:38:05 1989
Received: from Q.CS.CMU.EDU by B.GP.CS.CMU.EDU; 18 Jan 89 13:37:55 EST
Received: from CS.CMU.EDU by Q.CS.CMU.EDU; 17 Jan 89 22:52:50 EST
Received: from RICE-CHEX.AI.MIT.EDU by CS.CMU.EDU; 17 Jan 89 22:51:29 EST
Received: by rice-chex.ai.mit.edu; Tue, 17 Jan 89 22:47:17 EST
Date: Tue, 17 Jan 89 22:47:17 EST
From: poggio@wheaties.ai.mit.edu (Tomaso Poggio)
Message-Id: <8901180347.AA21088@rice-chex.ai.mit.edu>
To: sontag@fermat.rutgers.edu
Cc: Connectionists@cs.cmu.edu, rui@zeta.rice.edu
In-Reply-To: sontag@fermat.rutgers.edu's message of Tue, 17 Jan 89 14:08:03 EST <8901171908.AA00964@control.rutgers.edu>
Subject: Kolmogorov's superposition theorem

Kolmogorov 's theorem and its relation to networks are discussed in Biol. Cyber., 37, 167-186, 1979. (On the representation of multi-input systems: computational properties of polynomial algorithms. Poggio and Reichardt). There are references there to older papers (see especially the two nice papers by H. Abelson).

==========

<radish: 12> egrep "Kolmogorov" * >> ~/project/early-stats

arch.8901:Subject: Kolmogorov's superposition theorem
arch.8901:The implementation of the Kolmogorov-Arnold-Sprecher Superposition Theorem
arch.8901:[1] A.K.Kolmogorov, "On the representation of continuous functions of several
arch.8901:[3] D.A.Sprecher, "An improvement in the superposition theorem of Kolmogorov,"
arch.8901:[4] Rui J.P.de Figueiredo, "Implications and applications of Kolmogorov's
arch.8901:[5] R.Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem."
arch.8901:Subject: Kolmogorov's superposition theorem
arch.8901:Kolmogorov 's theorem and its relation to networks are discussed in
arch.8901: Subject: Kolmogorov's superposition theorem
arch.8901: Kolmogorov's theorem and its relation to networks
arch.8901: Kolmogorov's superposition theorem" (IEEE Trans. on
arch.8903:paper on "Kolmogorov's Mapping Neural Network Theorem" (1987 INNS proceedings?).
arch.8906:The mistake in this line of reasoning is as follows: Using the Kolmogorov
arch.8906:Naturally, ideas based on Kolmogorov complexity are easiest to apply if we
arch.8906:>> The mistake in this line of reasoning is as follows: Using the Kolmogorov
arch.8906:Moreover, any discussion of Kolmogorov complexity vs. modeling errors
arch.8906:about the Kolmogorov-Chaitin view of complexity is that (in the limit) it
arch.8906:Subject: Kolmogorov-Chaitin complexity (was: Categorization and Supervision)
arch.8906: about the Kolmogorov-Chaitin view of complexity is that (in the limit) it

004-notes-using-smart-to-find-par Fri May 2 09:30:35 1997 4

arch.8910:using Kolmogorov-Chaitin algorithmic complexity applied to dynamical
arch.9001:Representation properties of networks: Kolmogorov's theorem is irrelevant
arch.9001:instructions (Kolmogorov). I know that only a truly random string is
arch.9001:The ideas of minimum description lenght (mdl), Kolmogorov-Chaitin complexity.
arch.9008:Kolmogorov complexity of an architectural description be useful as a
arch.9105:"Chaitin-Kolmogorov Complexity and Generalization in Neural
arch.9112:Kolmogorov's Theorem is Relevant
arch.9207:generalization (Vapnik-Chervonenkis dimension, Kolmogorov methods,
arch.9210:* Kolmogorov Complexity; Universal Prior Distribution; generating "hard"
arch.9210:* A scheme for generating compressible binary vectors motivated by Kolmogorov
arch.9301:On the Realization of a Kolmogorov Network
arch.9305:The Kolmogorov Signal processor. Prof. M.A. Lagunas, A. Prez, M. Najar, A. Pags. UPC
arch.9404:Solomonoff, Kolmogorov, Chaitin, Wallace, Rissanen, and others. The
arch.9404:Kolmogorov complexity, algorithmic information theory, generalized
arch.9404:Kolmogorov complexity, minimum message-length inference, the minimum
arch.9408:Subject: Kolmogorov complexity and generalization
arch.9408: of those based on Kolmogorov complexity and Solomonoff's
arch.9408: Levin complexity (a time-bounded generalization of Kolmogorov
arch.9408: simple'' neural networks with low Kolmogorov complexity and high
arch.9501:{\em Learnability of Kolmogorov-easy circuit expressions via queries},
arch.9501:{\em Kolmogorov numberings and minimal identification},
arch.9505:Learnability of Kolmogorov-Easy Circuit Expressions Via Queries
arch.9505:the case in which the circuit expressions are of low (time-bounded) Kolmogorov
arch.9505:of the results to various Kolmogorov complexity bounds is discussed.

008-preparing-data-sets-digest-fo Fri May 2 09:30:35 1997 1

```
{28-May-96 : Additional notes on how breaking up text into appropriate
segments. These notes supplement the description of the formatting
code. }


KAK: 22 August 1995

created datasets using filmus-l. forgot to document it. am now
following up in the same fashion with wisa.

with archives filmus-l, imse-l, and wisa, they arrive in chunks of
digest format through email. then an imail hook runs and concatenates
them all to one file. we need to run headersparser to get rid of the
mbox headers. note that this assumes the archives are in NOT-mbox
format, because the script looks for the pinhead to strip out.

<radish: 1> pwd
/home/kknowles/project/explorations
<radish: 2> cd ../archives-from-listserv/wisa/
<radish: 3> ls
total 160
-rw-r-----    1 bl011261   73982 Jul  5 14:46 wisa-archives
<radish: 4> headerparser wisa-archives just-archives
<radish: 5> ls
total 312
-rw-r--r--    1 bl011261   72192 Aug 23 15:34 just-archives
-rw-r-----    1 bl011261   73982 Jul  5 14:46 wisa-archives

okay, now that's done. now i'd like to create the new test data
directory, and parse this archive into separate numbered messages.

<radish: 6> mkdir /radish/070/projwisa
<radish: 8> mkdir /radish/070/projwisa/messages

so we have parse-digest which uses the line "^Date:" as a message
delimiter, and we have trim-digest, which removes the message separator
==========================================================(approx.)

<radish: 9> parse-digest just-archives /radish/070/projwisa/messages
<radish: 10> ls /radish/070/projwisa/messages
total 165
-rw-r--r--    1 kknowles    1193 Aug 23 15:40 00001.mbox
-rw-r--r--    1 kknowles    1335 Aug 23 15:40 00002.mbox
-rw-r--r--    1 kknowles     832 Aug 23 15:40 00003.mbox
-rw-r--r--    1 kknowles    1368 Aug 23 15:40 00004.mbox
-rw-r--r--    1 kknowles     861 Aug 23 15:40 00005.mbox
-rw-r--r--    1 kknowles    2216 Aug 23 15:40 00006.mbox
-rw-r--r--    1 kknowles    1563 Aug 23 15:40 00007.mbox
-rw-r--r--    1 kknowles    4039 Aug 23 15:40 00008.mbox
-rw-r--r--    1 kknowles    1857 Aug 23 15:40 00009.mbox
-rw-r--r--    1 kknowles    1057 Aug 23 15:40 00010.mbox
-rw-r--r--    1 kknowles    1163 Aug 23 15:40 00011.mbox
-rw-r--r--    1 kknowles    1397 Aug 23 15:40 00012.mbox
-rw-r--r--    1 kknowles    1799 Aug 23 15:40 00013.mbox
-rw-r--r--    1 kknowles     790 Aug 23 15:40 00014.mbox
-rw-r--r--    1 kknowles     874 Aug 23 15:40 00015.mbox
-rw-r--r--    1 kknowles     674 Aug 23 15:40 00016.mbox
-rw-r--r--    1 kknowles    1195 Aug 23 15:40 00017.mbox
-rw-r--r--    1 kknowles    2528 Aug 23 15:40 00018.mbox
-rw-r--r--    1 kknowles    2372 Aug 23 15:40 00019.mbox
-rw-r--r--    1 kknowles    2627 Aug 23 15:40 00020.mbox
-rw-r--r--    1 kknowles    1937 Aug 23 15:40 00021.mbox
-rw-r--r--    1 kknowles    1037 Aug 23 15:40 00022.mbox
-rw-r--r--    1 kknowles     633 Aug 23 15:40 00023.mbox
-rw-r--r--    1 kknowles     901 Aug 23 15:40 00024.mbox
-rw-r--r--    1 kknowles     991 Aug 23 15:40 00025.mbox
-rw-r--r--    1 kknowles    1854 Aug 23 15:40 00026.mbox
-rw-r--r--    1 kknowles    7846 Aug 23 15:40 00027.mbox
-rw-r--r--    1 kknowles    1008 Aug 23 15:40 00028.mbox
-rw-r--r--    1 kknowles    1388 Aug 23 15:40 00029.mbox
-rw-r--r--    1 kknowles    1152 Aug 23 15:40 00030.mbox
-rw-r--r--    1 kknowles    2198 Aug 23 15:40 00031.mbox
-rw-r--r--    1 kknowles     864 Aug 23 15:40 00032.mbox
-rw-r--r--    1 kknowles     997 Aug 23 15:40 00033.mbox
-rw-r--r--    1 kknowles    1128 Aug 23 15:40 00034.mbox
-rw-r--r--    1 kknowles    1031 Aug 23 15:40 00035.mbox
-rw-r--r--    1 kknowles     908 Aug 23 15:40 00036.mbox
-rw-r--r--    1 kknowles     695 Aug 23 15:40 00037.mbox
-rw-r--r--    1 kknowles     511 Aug 23 15:40 00038.mbox
-rw-r--r--    1 kknowles    1239 Aug 23 15:40 00039.mbox
-rw-r--r--    1 kknowles     843 Aug 23 15:40 00040.mbox
-rw-r--r--    1 kknowles     755 Aug 23 15:40 00041.mbox
-rw-r--r--    1 kknowles     572 Aug 23 15:40 00042.mbox
-rw-r--r--    1 kknowles     425 Aug 23 15:40 00043.mbox
-rw-r--r--    1 kknowles    2097 Aug 23 15:40 00044.mbox
-rw-r--r--    1 kknowles    1334 Aug 23 15:40 00045.mbox
-rw-r--r--    1 kknowles     557 Aug 23 15:40 00046.mbox
-rw-r--r--    1 kknowles    2594 Aug 23 15:40 00047.mbox
-rw-r--r--    1 kknowles    1409 Aug 23 15:40 00048.mbox
-rw-r--r--    1 kknowles    1474 Aug 23 15:40 00049.mbox

<radish: 12> ls /radish/070/projwisa/messages
```

```
total 156
-rw-r--r--    1 kknowles    1119 Aug 23 15:42 00001.mbox
-rw-r--r--    1 kknowles    1261 Aug 23 15:42 00002.mbox
-rw-r--r--    1 kknowles     758 Aug 23 15:42 00003.mbox
-rw-r--r--    1 kknowles    1294 Aug 23 15:42 00004.mbox
-rw-r--r--    1 kknowles     787 Aug 23 15:42 00005.mbox
-rw-r--r--    1 kknowles    2142 Aug 23 15:42 00006.mbox
-rw-r--r--    1 kknowles    1489 Aug 23 15:42 00007.mbox
-rw-r--r--    1 kknowles    3965 Aug 23 15:42 00008.mbox
-rw-r--r--    1 kknowles    1783 Aug 23 15:42 00009.mbox
-rw-r--r--    1 kknowles     983 Aug 23 15:42 00010.mbox
-rw-r--r--    1 kknowles    1089 Aug 23 15:42 00011.mbox
-rw-r--r--    1 kknowles    1323 Aug 23 15:42 00012.mbox
-rw-r--r--    1 kknowles    1725 Aug 23 15:42 00013.mbox
-rw-r--r--    1 kknowles     716 Aug 23 15:42 00014.mbox
-rw-r--r--    1 kknowles     800 Aug 23 15:42 00015.mbox
-rw-r--r--    1 kknowles     600 Aug 23 15:42 00016.mbox
-rw-r--r--    1 kknowles    1121 Aug 23 15:42 00017.mbox
-rw-r--r--    1 kknowles    2454 Aug 23 15:42 00018.mbox
-rw-r--r--    1 kknowles    2298 Aug 23 15:42 00019.mbox
-rw-r--r--    1 kknowles    2553 Aug 23 15:42 00020.mbox
-rw-r--r--    1 kknowles    1863 Aug 23 15:42 00021.mbox
-rw-r--r--    1 kknowles     963 Aug 23 15:42 00022.mbox
-rw-r--r--    1 kknowles     559 Aug 23 15:42 00023.mbox
-rw-r--r--    1 kknowles     827 Aug 23 15:42 00024.mbox
-rw-r--r--    1 kknowles     917 Aug 23 15:42 00025.mbox
-rw-r--r--    1 kknowles    1780 Aug 23 15:42 00026.mbox
-rw-r--r--    1 kknowles    7772 Aug 23 15:42 00027.mbox
-rw-r--r--    1 kknowles     934 Aug 23 15:42 00028.mbox
-rw-r--r--    1 kknowles    1314 Aug 23 15:42 00029.mbox
-rw-r--r--    1 kknowles    1078 Aug 23 15:42 00030.mbox
-rw-r--r--    1 kknowles    2124 Aug 23 15:42 00031.mbox
-rw-r--r--    1 kknowles     790 Aug 23 15:42 00032.mbox
-rw-r--r--    1 kknowles     923 Aug 23 15:42 00033.mbox
-rw-r--r--    1 kknowles    1054 Aug 23 15:42 00034.mbox
-rw-r--r--    1 kknowles     957 Aug 23 15:42 00035.mbox
-rw-r--r--    1 kknowles     834 Aug 23 15:42 00036.mbox
-rw-r--r--    1 kknowles     621 Aug 23 15:42 00037.mbox
-rw-r--r--    1 kknowles     417 Aug 23 15:42 00038.mbox
-rw-r--r--    1 kknowles    1165 Aug 23 15:42 00039.mbox
-rw-r--r--    1 kknowles     769 Aug 23 15:42 00040.mbox
-rw-r--r--    1 kknowles     681 Aug 23 15:42 00041.mbox
-rw-r--r--    1 kknowles     498 Aug 23 15:42 00042.mbox
-rw-r--r--    1 kknowles     351 Aug 23 15:42 00043.mbox
-rw-r--r--    1 kknowles    2023 Aug 23 15:42 00044.mbox
-rw-r--r--    1 kknowles    1334 Aug 23 15:42 00045.mbox
-rw-r--r--    1 kknowles     483 Aug 23 15:42 00046.mbox
-rw-r--r--    1 kknowles    2520 Aug 23 15:42 00047.mbox
-rw-r--r--    1 kknowles    1335 Aug 23 15:42 00048.mbox
-rw-r--r--    1 kknowles    1474 Aug 23 15:42 00049.mbox
```

okay, so we see we have 49 messages.

this done. we copy over the create_full_files file from somewhere, like /radish/070/projfilm1/.

```
<radish: 14> cd /radish/070/projwisa
<radish: 15> cp  ./projfilm1/create_full_files ./
<radish: 16> ls
total 11
-rwxr-xr-x    1 kknowles    4303 Aug 23 15:48 create_full_files*
drwxr-xr-x    2 kknowles    1024 Aug 23 15:42 messages/
```

and we alter create_full_files for this set. the only place that needs to become personalized is the location of the smart databases. of course, this all becomes irrelevant when we realize we have little place to get the 'answers' or the 'truth' from. but we can try.

```
<radish: 17> create_full_files 2> cff-log
Wed Aug 23 15:52:21 EDT 1995
Wed Aug 23 15:52:22 EDT 1995
/radish/070/projwisa
Indexing docs at Wed Aug 23 15:52:23 EDT 1995
0.3u 2.6s 0:04 61% 0+0k 71+12io 53pf+0w
All done at Wed Aug 23 15:52:27 EDT 1995
Wed Aug 23 15:52:27 EDT 1995
/radish/070/projwisa
Indexing docs at Wed Aug 23 15:52:28 EDT 1995
0.2u 0.6s 0:01 64% 0+0k 4+12io 1pf+0w
All done at Wed Aug 23 15:52:28 EDT 1995
Wed Aug 23 15:52:29 EDT 1995
/radish/070/projwisa
Indexing docs at Wed Aug 23 15:52:29 EDT 1995
0.3u 2.5s 0:03 85% 0+0k 5+12io 0pf+0w
All done at Wed Aug 23 15:52:32 EDT 1995
Wed Aug 23 15:52:32 EDT 1995
Wed Aug 23 15:52:32 EDT 1995
Wed Aug 23 15:52:33 EDT 1995
Wed Aug 23 15:52:33 EDT 1995
Wed Aug 23 15:52:33 EDT 1995
Wed Aug 23 15:52:33 EDT 1995
Wed Aug 23 15:52:34 EDT 1995
```

```
Wed Aug 23 15:52:34 EDT 1995
Wed Aug 23 15:52:34 EDT 1995
Wed Aug 23 15:52:34 EDT 1995
Wed Aug 23 15:52:35 EDT 1995
Wed Aug 23 15:52:36 EDT 1995
Wed Aug 23 15:52:36 EDT 1995
Wed Aug 23 15:52:36 EDT 1995
Wed Aug 23 15:52:36 EDT 1995
Wed Aug 23 15:52:36 EDT 1995
Wed Aug 23 15:52:37 EDT 1995

okay, so let's look at what we've got:

<radish: 18> pwd
/radish/070/projwisa
<radish: 19> ls
total 27
-rw-r--r--    1 kknowles    1354 Aug 23 15:52 cff-log
-rw-r--r--    1 kknowles       0 Aug 23 15:52 child-parent-pairs-man-id
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-qq-qu
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-qq-qu-uq
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-qq-qu-uq-uu
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-qq-qu-uq-uu-s
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-quotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-quotq-unquot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-subj
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-unquotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 cprs-unquotq-unquot
-rwxr-xr-x    1 kknowles    4301 Aug 23 15:50 create_full_files*
-rwxr-xr-x    1 kknowles    4303 Aug 23 15:48 create_full_files-*
-rw-r--r--    1 kknowles       0 Aug 23 15:52 full-parent-rank-quotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 full-parent-rank-quotq-unquot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 full-parent-rank-subj
-rw-r--r--    1 kknowles       0 Aug 23 15:52 full-parent-rank-unquotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 full-parent-rank-unquotq-unquot
drwxr-xr-x    2 kknowles    1024 Aug 23 15:42 messages/
-rw-r--r--    1 kknowles       0 Aug 23 15:52 num-retrieved-quotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 num-retrieved-quotq-unquot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 num-retrieved-subj
-rw-r--r--    1 kknowles       0 Aug 23 15:52 num-retrieved-unquotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 num-retrieved-unquotq-unquot
drwxr-xr-x    2 kknowles    1024 Aug 23 15:52 quotes/
-rw-r--r--    1 kknowles       0 Aug 23 15:52 rp-table-quotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 rp-table-quotq-unquot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 rp-table-subj
-rw-r--r--    1 kknowles       0 Aug 23 15:52 rp-table-unquotq-quot
-rw-r--r--    1 kknowles       0 Aug 23 15:52 rp-table-unquotq-unquot
drwxr-xr-x    2 kknowles    1024 Aug 23 15:52 subjects/

which means that basically, i think everything is working, only there
are no irt identifications, so there is no data to be had.  we could,
however, run everything against everything else, old style, using
query-run-better.

<radish: 62> query-run-better quotes/ /radish/070/kim-email-dbases/projwisaunquot/spec 50 full-query-quotq-unquot

<radish: 64> query-run-better quotes/ /radish/070/kim-email-dbases/projwisaquot/spec 50 full-query-quotq-quot

which works just fine: so maybe with the digest case, we just want to look at this.  we
can definitely work with the entire wisa archive, and maybe we can test on the other

<radish: 117> create_full_files 2> cff-log
Wed Aug 23 18:02:23 EDT 1995
Wed Aug 23 18:02:24 EDT 1995
/radish/070/projwisa
Indexing docs at Wed Aug 23 18:02:25 EDT 1995
0.3u 2.4s 0:03 69% 0+0k 9+31io 2pf+0w
All done at Wed Aug 23 18:02:27 EDT 1995
Wed Aug 23 18:02:28 EDT 1995
/radish/070/projwisa
Indexing docs at Wed Aug 23 18:02:29 EDT 1995
0.6u 2.6s 0:04 79% 0+0k 5+32io 0pf+0w
All done at Wed Aug 23 18:02:32 EDT 1995
Wed Aug 23 18:02:32 EDT 1995
/radish/070/projwisa
Indexing docs at Wed Aug 23 18:02:33 EDT 1995
0.3u 2.5s 0:03 72% 0+0k 3+33io 0pf+0w
All done at Wed Aug 23 18:02:36 EDT 1995
Wed Aug 23 18:02:36 EDT 1995
Wed Aug 23 18:02:52 EDT 1995
Wed Aug 23 18:02:52 EDT 1995
Wed Aug 23 18:03:08 EDT 1995
Wed Aug 23 18:03:08 EDT 1995
Wed Aug 23 18:03:39 EDT 1995
Wed Aug 23 18:03:39 EDT 1995
Wed Aug 23 18:04:12 EDT 1995
Wed Aug 23 18:04:13 EDT 1995
Wed Aug 23 18:04:41 EDT 1995
Wed Aug 23 18:04:42 EDT 1995
Wed Aug 23 18:04:43 EDT 1995
Wed Aug 23 18:04:43 EDT 1995
Wed Aug 23 18:04:43 EDT 1995
```

```
Wed Aug 23 18:04:44 EDT 1995
Wed Aug 23 18:04:44 EDT 1995
Wed Aug 23 18:04:46 EDT 1995
<radish: 118> ls
total 231
-rw-r--r--    1 kknowles     696 Aug 23 18:04 cff-log
drwxr-xr-x    2 kknowles     512 Aug 23 18:04 cprs-concat/
-rw-r--r--    1 kknowles  106095 Aug 23 18:04 cprs-final
drwxr-xr-x    2 kknowles     512 Aug 23 18:04 cprs-origs/
-rwxr-xr-x    1 kknowles    4349 Aug 23 18:01 create_full_files*
drwxr-xr-x    2 kknowles     512 Aug 23 18:04 full-queries/
drwxr-xr-x    2 kknowles    1024 Aug 23 15:42 messages/
drwxr-xr-x    2 kknowles     512 Aug 23 18:04 num-rets/
drwxr-xr-x    2 kknowles    1024 Aug 23 18:02 quotes/
drwxr-xr-x    2 kknowles    1024 Aug 23 18:02 subjects/
drwxr-xr-x    2 kknowles    1024 Aug 23 18:02 unquotes/

so we're done.
```

```
#! /bin/csh
# 28-May-96 : Code which drives SMART to set up the database of
# quoted material.

# usage: make_text_uquot collection dbase

# set echo verbose

#
#set bin = /home/smart/src/bin
set bin = /home/lewis/public/sgi/bin
#set tlibdir = /home/smart/lib
set tlibdir = /home/lewis/public/src/smart/smart.11.0/lib
#set database = /home/smart/indexed_colls/text
#set database = /radish/070/kim-email-dbases/projmailsubj
#set database = /radish/070/kim-email-dbases/projmailquot
#set database = /radish/070/kim-email-dbases/projtestquot
#set database = /radish/070/kim-email-dbases/projtest10kquot
set database = $2
#set coll     = $cwd
#set coll     = /radish/070/projtrain/subjects/
#set coll     = /radish/070/projtrain/quotes/
#set coll     = /radish/070/projtest10k/quotes/
set coll     = $1

# create the empty collection (destroying any existing collection)
/bin/rm -rf $database
mkdir $database

find $coll -type f -print > $database/doc_loc
# for some reason, the ls method didn't work....
#/bin/ls -1 /radish/070/projtrain/subjects/ > $database/doc_loc

cat > $database/spec << EOF1
## INFORMATION LOCATIONS
database                $database
include_file            $tlibdir/spec.default

## TEXT DOCDESC
#### GENERIC PREPARSER
pp.default_section_name         w
pp.default_section_action       copy

#### DESCRIPTION OF PARSE INPUT
index.num_sections              1
index.section.0.name            w
  index.section.0.method        index.parse_sect.full
  index.section.0.word.ctype    0
  index.section.0.proper.ctype  0
#ctype.0.text_stop_file --      #we want to maintain the stop word list for quoted material
title_section 0

#### DESCRIPTION OF FINAL VECTORS
num_ctypes                      1

## ALTERATIONS OF STANDARD PARAMETERS
doc_weight                      nnc
query_weight                    ntc

## ALTERATIONS OF STANDARD PROCEDURES
EOF1

#index the collection
echo Indexing docs at `date`
$bin/smart index.doc $database/spec < $database/doc_loc

time
echo All done at `date`
```

make_text_subj Fri May 2 09:31:01 1997 1

```
#! /bin/csh
# 28-May-96 : Code which drives SMART to set up the database of
# subject lines.

# usage: make_text_subj collection dbase

# set echo verbose

#
#set bin =      /home/smart/src/bin
set bin = /home/lewis/public/sgi/bin
#set tlibdir = /home/smart/lib
set tlibdir = /home/lewis/public/src/smart/smart.11.0/lib
#set database = /home/smart/indexed_colls/text
#set database = /radish/070/kim-email-dbases/projmailsubj
#set database = /radish/070/kim-email-dbases/projtestsubj
#set database = /radish/070/kim-email-dbases/projtest10ksubj
set database = $2
#set coll      = $cwd
#set coll      = /radish/070/projtrain/subjects/
#set coll      = /radish/070/projtest10k/subjects/
set coll      = $1

# create the empty collection (destroying any existing collection)
/bin/rm -rf $database
mkdir $database

find $coll -type f -print > $database/doc_loc
# for some reason, the ls method didn't work....
#/bin/ls -1 /radish/070/projtrain/subjects/ > $database/doc_loc

cat > $database/spec << EOF1
## INFORMATION LOCATIONS
database                $database
include_file            $tlibdir/spec.default

## TEXT DOCDESC
#### GENERIC PREPARSER
pp.default_section_name         w
pp.default_section_action       copy

#### DESCRIPTION OF PARSE INPUT
index.num_sections              1
index.section.0.name            w
  index.section.0.method        index.parse_sect.full
  index.section.0.word.ctype    0
  index.section.0.proper.ctype  0
ctype.0.text_stop_file  --      # we don't want a stop word list on the subject
title_section 1

#### DESCRIPTION OF FINAL VECTORS
num_ctypes                      1

## ALTERATIONS OF STANDARD PARAMETERS
doc_weight                      nnc
query_weight                    ntc

## ALTERATIONS OF STANDARD PROCEDURES
EOF1

#index the collection
echo Indexing docs at `date`
$bin/smart index.doc $database/spec < $database/doc_loc

time
echo All done at `date`
```

```
#! /bin/csh
# 28-May-96 : Code which drives SMART to set up the database of
# unquoted material.

# usage: make_text_uquot collection dbase

# set echo verbose

#
#set bin =      /home/smart/src/bin
set bin = /home/lewis/public/sgi/bin
#set tlibdir = /home/smart/lib
set tlibdir = /home/lewis/public/src/smart/smart.11.0/lib
#set database = /home/smart/indexed_colls/text
#set database = /radish/070/kim-email-dbases/projmailsubj
#set database = /radish/070/kim-email-dbases/projmailunquot
#set database = /radish/070/kim-email-dbases/projtestunquot
#set database = /radish/070/kim-email-dbases/projtest10kunquot
set database = $2
#set coll      = $cwd
#set coll      = /radish/070/projtrain/subjects/
#set coll      = /radish/070/projtrain/unquotes/
#set coll      = /radish/070/projtest10k/unquotes/
set coll      = $1

# create the empty collection (destroying any existing collection)
/bin/rm -rf $database
mkdir $database

find $coll -type f -print > $database/doc_loc
# for some reason, the ls method didn't work....
#/bin/ls -1 /radish/070/projtrain/subjects/ > $database/doc_loc

cat > $database/spec << EOF1
## INFORMATION LOCATIONS
database                $database
include_file            $tlibdir/spec.default

## TEXT DOCDESC
#### GENERIC PREPARSER
pp.default_section_name         w
pp.default_section_action       copy

#### DESCRIPTION OF PARSE INPUT
index.num_sections              1
index.section.0.name            w
  index.section.0.method        index.parse_sect.full
  index.section.0.word.ctype    0
  index.section.0.proper.ctype  0
#ctype.0.text_stop_file --      # we want to use the stop list
title_section 0

#### DESCRIPTION OF FINAL VECTORS
num_ctypes                      1

## ALTERATIONS OF STANDARD PARAMETERS
doc_weight                      nnc
query_weight                    ntc

## ALTERATIONS OF STANDARD PROCEDURES
EOF1

#index the collection
echo Indexing docs at `date`
$bin/smart index.doc $database/spec < $database/doc_loc

time
echo All done at `date`
```

What is Claimed is:

1. A method of determining from a plurality of messages a second message that is related to a first message, comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. determining from each of the degrees of match which one of the plurality of messages is the second message.

2. The method according to claim 1, wherein the relationship of the second message to the first message is parent to child;

wherein the first message filter bank comprises a message filter that extracts a quoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered.

3. The method according to claim 1, wherein the relationship of the second message to the first message is child to parent;

wherein the first message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts a quoted portion of the message being filtered.

4. The method according to claim 1, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector comprises use of a statistical information retrieval function.

5. The method according to claim 1, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which one of each of the degrees of match is the maximum value and selecting the message corresponding to the determined maximum value.

6. The method according to claim 4, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vectors further comprises combining a set of values resulting from the statistical information retrieval function to form a single value representative of the degree of match.

7. The method according to claim 6, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which element of the tuple of values representative of each of the degrees of match is the maximum value, and selecting the message corresponding to

the determined maximum value.

8. The method according to claim 1, further comprising the step of if the first message is contained in the plurality of messages, removing the first message from the plurality of messages before filtering the plurality of messages using the second message filter bank.

9. The method according to claim 1, further comprising the step of verifying that the second message is related to the first message.

10. The method according to claim 9, wherein the step of verifying that the second message is related to the first message includes determining whether the degree of match between the filtered first message vector and the filtered second message vector corresponding to the determined second message exceeds a threshold value.

11. The method according to claim 1, further comprising the step of presenting a list including the first message, at least one of the plurality of messages, and the degree of match between the filtered first message vector and the filtered second message vector corresponding to the at least one of the plurality of messages.

12. A method of determining from a plurality of messages whether a second message is related to a first message, comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. determining for each of the set of filtered second message vectors whether the degree of match between the filtered first message vector and the filtered second message vector exceeds a threshold value.

13. A method of processing a plurality of messages that may be related to a first message, comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. presenting a list including the first message, at least one of the plurality of messages, and the degree of match between the filtered first message vector and the filtered second message vector corresponding to the at least one of the plurality of messages.

14. A method of determining a thread of related messages from a plurality of messages, comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. if the first message is contained in the plurality of messages, removing the first message from the plurality of messages;

c. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

d. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector;

e. determining from each of the degrees of match whether one of the plurality of messages is a second message related to the first message; and

f. if it is determined that one of plurality of messages is a second message is related to the first message, substituting the second message in place of the first message and repeating each of the steps a through f herein.

15. The method according to claim 14, wherein the relationship of the second message to the first message is parent to child;

wherein the first message filter bank comprises a message filter that extracts a quoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered.

16. The method according to claim 14, wherein the relationship of the second message to the first message is child to parent;

wherein the first message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts a quoted portion of the message being filtered.

17. The method according to claim 14, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector comprises use of a statistical information retrieval function.

18. The method according to claim 14, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which one of each of the degrees of match is the maximum value and selecting the message corresponding to the determined maximum value.

19. The method according to claim 17, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector further comprises combining a set of values resulting from the statistical information retrieval function to form a single value representative of the degree of match.

20. The method according to claim 19, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which element of the vector representative of each of the degrees of match is the maximum value, and selecting the message corresponding to the determined maximum value.

21. A method of determining a thread of related messages from a plurality of messages, comprising the steps of:

a. generating a set of filtered first message vectors by filtering each of the plurality of messages using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between each of the filtered first message vectors and the filtered second message vector;

d. determining from each of the degrees of match each one of the plurality of messages that is related to another of the plurality of messages; and

e. determining from each of the plurality of messages that is related to another of the plurality of messages a linked list of messages having successive parent-child relationships.

22. A system for determining from a plurality of messages a second message that is related to a first

message, comprising:

a. a processor; and

b. memory;

wherein said processor is programmed to execute the steps of:

1. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

2. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

3. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

4. determining from each of the degrees of match which one of the plurality of messages is the second message.

23. The system according to claim 22, wherein the relationship of the second message to the first message is parent to child;

wherein the first message filter bank comprises a message filter that extracts a quoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered.

24. The system according to claim 22, wherein the relationship of the second message to the first message is child to parent;

wherein the first message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts a quoted portion of the message being filtered.

25. The system according to claim 22, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector comprises use of a statistical information retrieval function.

26. The system according to claim 22, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which one of each of the degrees of match is the maximum value and selecting the message corresponding to the determined maximum value.

27. The system according to claim 25, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector further comprises combining a set of values resulting from the statistical information retrieval function to form a single value representative of the degree of match.

28. The system according to claim 27, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which element of the tuple of values representative of each of the degrees of match is the maximum value, and selecting the message corresponding to

the determined maximum value.

29. The system according to claim 22, further comprising the step of if the first message is contained in the plurality of messages, removing the first message from the plurality of messages before filtering the plurality of messages using the second message filter bank.

30. The system according to claim 22, further comprising the step of verifying that the second message is related to the first message.

31. The system according to claim 30, wherein the step of verifying that the second message is related to the first message includes determining whether the degree of match between the filtered first message vector and the filtered second message vector corresponding to the determined second message exceeds a threshold value.

32. The system according to claim 22, further comprising the step of presenting a list including the first message, at least one of the plurality of messages, and the degree of match between the filtered first message vector and the filtered second message vector corresponding to the at least one of the plurality of messages.

33. A system for determining from a plurality of messages whether a second message is related to a first message, comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. determining for each of the set of filtered second message vectors whether the degree of match between the filtered first message vector and the filtered second message vector exceeds a threshold value.

34. A system for processing a plurality of messages that may be related to a first message, comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. presenting a list including the first message, at least one of the plurality of messages, and the degree of match between the filtered first message vector and the filtered second message vector corresponding to the at least one of the plurality of messages.

35. A system for determining a thread of related messages from a plurality of messages, comprising:

a. a processor; and

b. memory;

wherein said processor is programmed to execute the steps of:

1. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

2. if the first message is contained in the plurality of messages, removing the first message from the plurality of messages;

3. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

4. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector;

5. determining from each of the degrees of match whether one of the plurality of messages is a second message related to the first message; and

6. if it is determined that one of plurality of messages is a second message is related to the first message, substituting the second message in place of the first message and repeating each of the steps a through f herein.

36. The system according to claim 35, wherein the relationship of the second message to the first message is parent to child;

wherein the first message filter bank comprises a message filter that extracts a quoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered.

37. The system according to claim 35, wherein the relationship of the second message to the first message is child to parent;

wherein the first message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts a quoted portion of the message being filtered.

38. The system according to claim 35, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector comprises use of a statistical information retrieval function.

39. The system according to claim 35, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which one of each of the degrees of match is the maximum value and selecting the message corresponding to the determined maximum value.

40. The system according to claim 38, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vectors further comprises combining a set of values

resulting from the statistical information retrieval function to form a single value representative of the degree of match.

41. The system according to claim 40, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which element of the vector representative of each of the degrees of match is the maximum value, and selecting the message corresponding to the determined maximum value.

42. A system for determining a thread of related messages from a plurality of messages, comprising the steps of:

a. generating a set of filtered first message vectors by filtering each of the plurality of messages using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between each of the filtered first message vectors and the filtered second message vector;

d. determining from each of the degrees of match each one of the plurality of messages that is related to another of the plurality of messages; and

e. determining from each of the plurality of messages that is related to another of the plurality of messages a linked list of messages having successive parent-child relationships.

43. An article of manufacture, comprising a computer-readable medium having stored thereon instructions for determining from a plurality of messages a second message that is related to a first message, said instructions which, when performed by a processor, cause the processor to execute the steps comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. determining from each of the degrees of match which one of the plurality of messages is the second message.

44. The article of manufacture according to claim 43, wherein the relationship of the second message to the first message is parent to child;

wherein the first message filter bank comprises a message filter that extracts a quoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered.

45. The article of manufacture according to claim 43, wherein the relationship of the second message to the first message is child to parent;

wherein the first message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts a quoted portion of the message being filtered

46. The article of manufacture according to claim 43, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector comprises use of a statistical information retrieval function.

47. The article of manufacture according to claim 43, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which one of each of the degrees of match is the maximum value and selecting the message corresponding to the determined maximum value.

48. The article of manufacture according to claim 46, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector further comprises combining a set of values resulting from the statistical information retrieval function to form a single value representative of the degree of match.

49. The article of manufacture according to claim 48, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second

message comprises determining which element of the tuple of values representative of each of the degrees of match is the maximum value, and selecting the message corresponding to the determined maximum value.

50. The article of manufacture according to claim 43, further comprising the step of if the first message is contained in the plurality of messages, removing the first message from the plurality of messages before filtering the plurality of messages using the second message filter bank.

51. The article of manufacture according to claim 43, further comprising the step of verifying that the second message is related to the first message.

52. The article of manufacture according to claim 51, wherein the step of verifying that the second message is related to the first message includes determining whether the degree of match between the filtered first message vector and the filtered second message vector corresponding to the determined second message exceeds a threshold value.

53. The article of manufacture according to claim 43, further comprising the step of presenting a list including the first message, at least one of the plurality of messages, and the degree of match between the filtered first message vector and the filtered second message vector corresponding to the at least one of the plurality of messages.

54. An article of manufacture comprising a computer-readable medium having stored thereon instructions for determining from a plurality of messages whether a second message is related to a first message, said instructions

which, when performed by a processor, cause the processor to execute the steps comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. determining for each of the set of filtered second message vectors whether the degree of match between the filtered first message vector and the filtered second message vector exceeds a threshold value.

55. An article of manufacture comprising a computer-readable medium having stored thereon instructions for processing a plurality of messages that may be related to a first message, said instructions which, when performed by a processor, cause the processor to execute the steps comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector; and

d. presenting a list including the first message, at least one of the plurality of messages, and the degree of match between the filtered first message vector and the filtered second message vector corresponding to the at least one of the plurality of messages.

56. An article of manufacture, comprising a computer-readable medium having stored thereon instructions for determining a thread of related messages from a plurality of messages, said instructions which, when performed by a processor, cause the processor to execute the steps comprising the steps of:

a. generating a filtered first message vector by filtering the first message using a first message filter bank, said first message filter bank comprising at least one message filter;

b. if the first message is contained in the plurality of messages, removing the first message from the plurality of messages;

c. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message filter bank comprising at least one message filter;

d. determining for each of the set of filtered second message vectors the degree of match between the filtered first message vector and the filtered second message vector;

e. determining from each of the degrees of match whether one of the plurality of messages is a second message related to the first message; and

f. if it is determined that one of plurality of messages is a second message is related to the first message, substituting the second message in place of the first message and repeating each of the steps a through f herein.

57. The article of manufacture according to claim 56, wherein the relationship of the second message to the first message is parent to child;

wherein the first message filter bank comprises a message filter that extracts a quoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered.

58. The article of manufacture according to claim 56, wherein the relationship of the second message to the first message is child to parent;

wherein the first message filter bank comprises a message filter that extracts an unquoted portion of the message being filtered; and

wherein the second message filter bank comprises a message filter that extracts a quoted portion of the message being filtered.

59. The article of manufacture according to claim 56, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector comprises use of a statistical information retrieval function.

60. The article of manufacture according to claim 56, wherein the step of determining from each of the degrees of

match which one of the plurality of messages is the second message comprises determining which one of each of the degrees of match is the maximum value and selecting the message corresponding to the determined maximum value.

61. The article of manufacture according to claim 59, wherein the step of determining the degree of match between the filtered first message vector and the filtered second message vector further comprises combining a set of values resulting from the statistical information retrieval function to form a single value representative of the degree of match.

62. The article of manufacture according to claim 61, wherein the step of determining from each of the degrees of match which one of the plurality of messages is the second message comprises determining which element of the vector representative of each of the degrees of match is the maximum value, and selecting the message corresponding to the determined maximum value.

63. An article of manufacture comprising a computer-readable medium having stored thereon instructions for determining a thread of related messages from a plurality of messages, said instructions which, when performed by a processor, cause the processor to execute the steps comprising the steps of:

a. generating a set of filtered first message vectors by filtering each of the plurality of messages using a first message filter bank, said first message filter bank comprising at least one message filter;

b. generating a set of filtered second message vectors by filtering each of the plurality of messages using a second message filter bank, said second message

filter bank comprising at least one message filter;

c. determining for each of the set of filtered second message vectors the degree of match between each of the filtered first message vectors and the filtered second message vector;

d. determining from each of the degrees of match each one of the plurality of messages that is related to another of the plurality of messages; and

e. determining from each of the plurality of messages that is related to another of the plurality of messages a linked list of messages having successive parent-child relationships.

FIG. 1

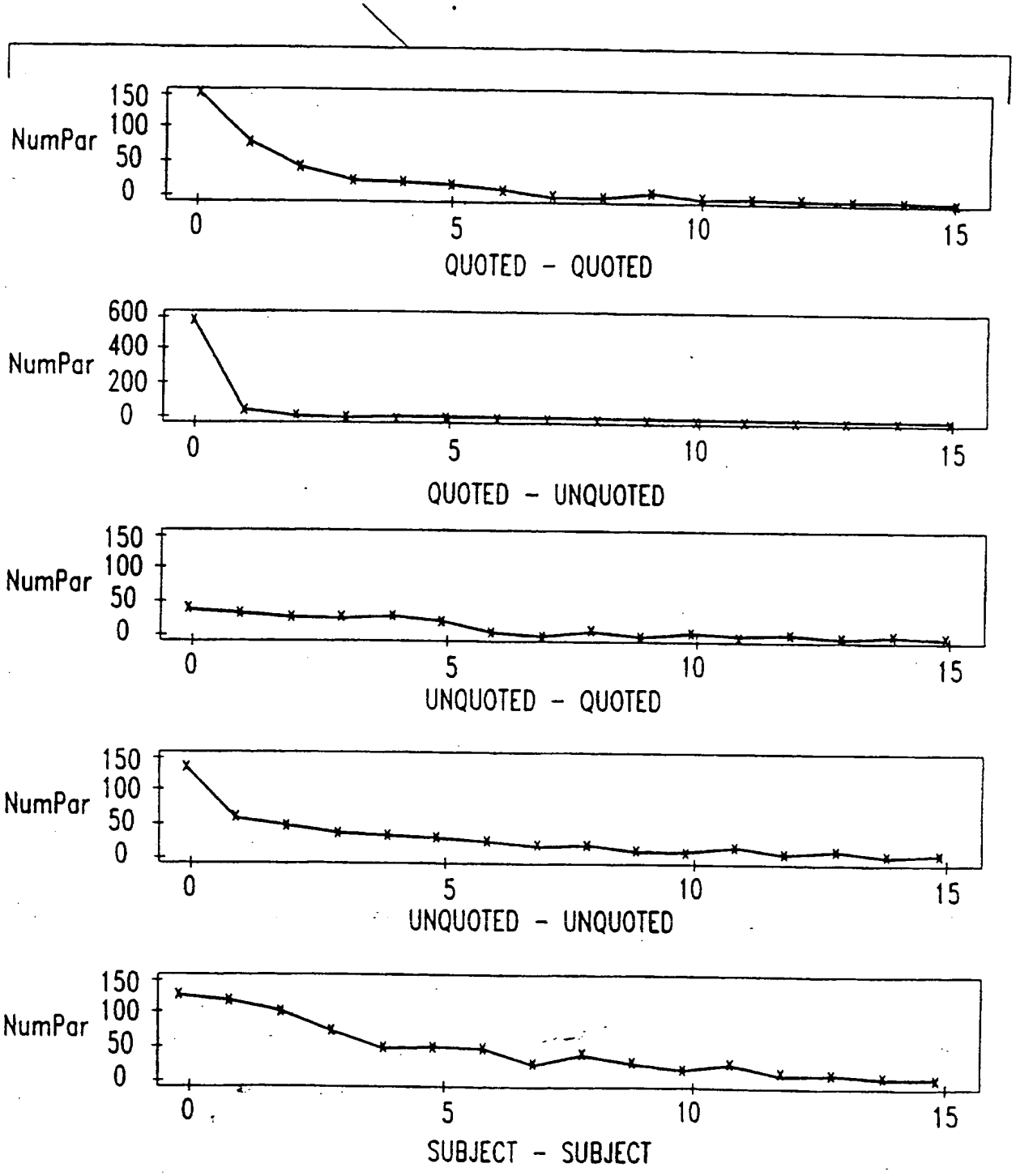

NumPar
150
100
50
0
0
5
10
15
QUOTED - QUOTED
NumPar
600
400
200
0
0
5
10
15
QUOTED - UNQUOTED
NumPar
150
100
50
0
0
5
10
15
UNQUOTED - QUOTED
NumPar
150
100
50
0
0
5
10
15
UNQUOTED - UNQUOTED
NumPar
150
100
50
0
0
5
10
15
SUBJECT - SUBJECT

FIG. 2

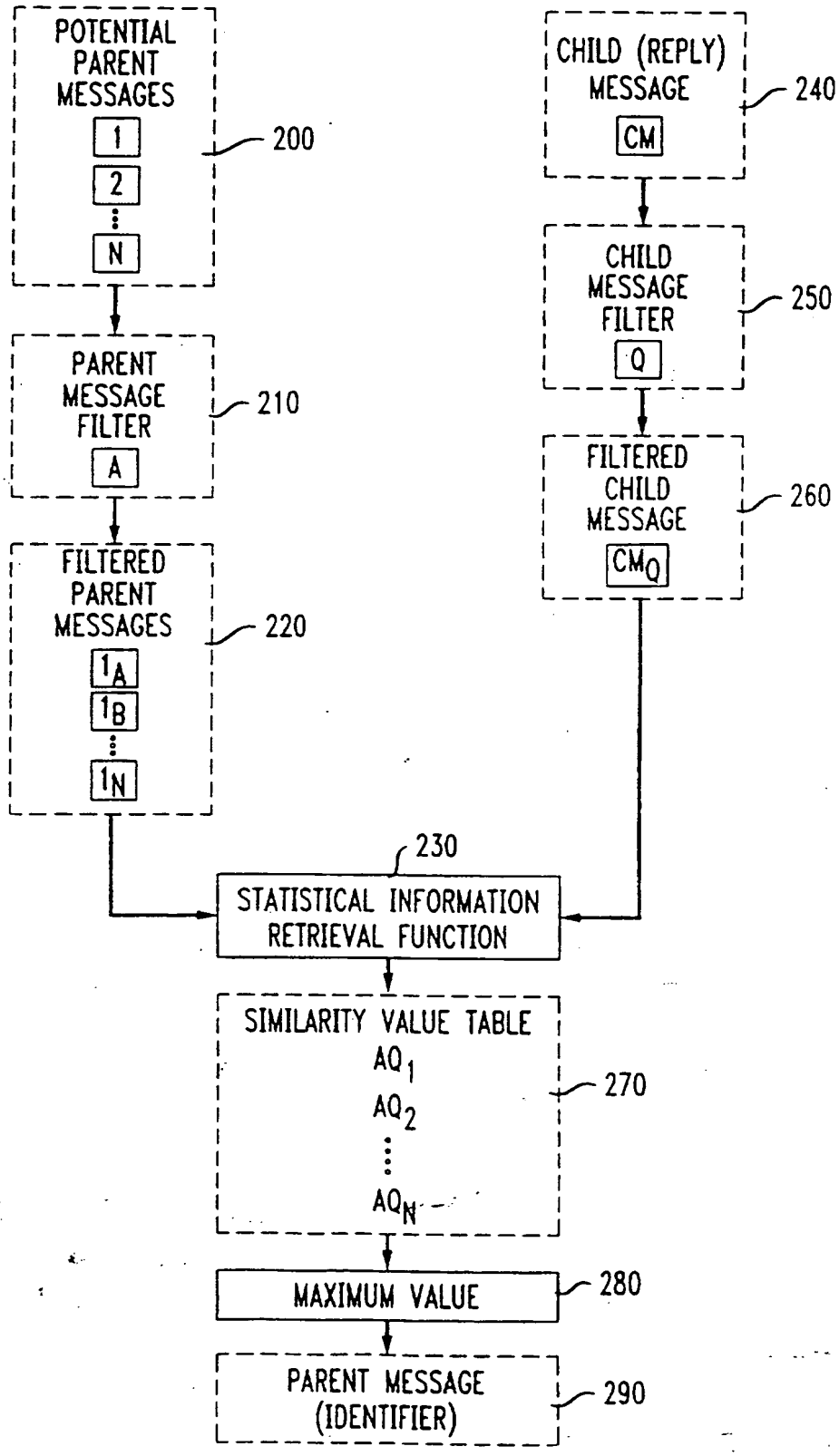
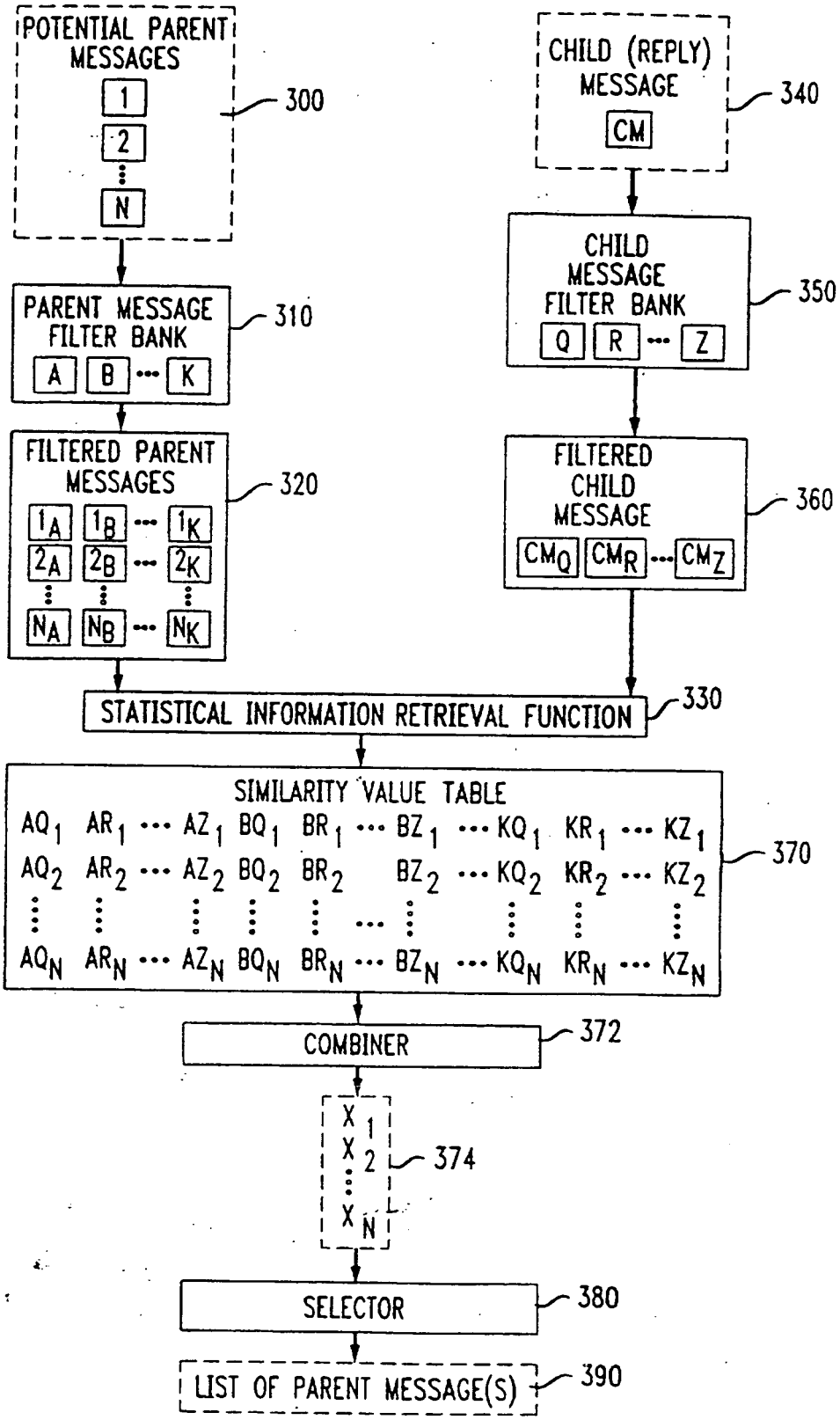

POTENTIAL PARENT MESSAGES
1
2
N
200
PARENT MESSAGE FILTER
A
210
FILTERED PARENT MESSAGES
1A
1B
1N
220
CHILD (REPLY) MESSAGE
CM
240
CHILD MESSAGE FILTER
Q
250
FILTERED CHILD MESSAGE
CMQ
260
230
STATISTICAL INFORMATION RETRIEVAL FUNCTION
SIMILARITY VALUE TABLE
AQ1
AQ2
AQN
270
MAXIMUM VALUE
280
PARENT MESSAGE (IDENTIFIER)
290

FIG. 3


POTENTIAL PARENT MESSAGES
1
2
N
300
CHILD (REPLY) MESSAGE
CM
340
PARENT MESSAGE FILTER BANK
A B ··· K
310
CHILD MESSAGE FILTER BANK
Q R ··· Z
350
FILTERED PARENT MESSAGES
1A 1B ··· 1K
2A 2B ··· 2K
NA NB ··· NK
320
FILTERED CHILD MESSAGE
CMQ CMR ··· CMZ
360
STATISTICAL INFORMATION RETRIEVAL FUNCTION
330
SIMILARITY VALUE TABLE
AQ1 AR1 ··· AZ1 BQ1 BR1 ··· BZ1 ··· KQ1 KR1 ··· KZ1
AQ2 AR2 ··· AZ2 BQ2 BR2 BZ2 ··· KQ2 KR2 ··· KZ2
AQN ARN ··· AZN BQN BRN ··· BZN ··· KQN KRN ··· KZN
370
COMBINER
372
X1
X2
XN
374
SELECTOR
380
LIST OF PARENT MESSAGE(S)
390

INTERNATIONAL SEARCH REPORT

Interr nal Application No
PCT/US 97/09161

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/60 G06F17/30 G06F17/20 H04L12/58

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SALTON G ET AL: "AUTOMATIC STRUCTURING AND RETRIEVAL OF LARGE TEXT FILES" COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY, vol. 37, no. 2, 1 February 1994, pages 97-108, XP000425939 see the whole document --- -/--	1-63

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
5 November 1997	18. 11. 97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Mikkelsen, C

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Interr nal Application No
PCT/US 97/09161

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	FREI H P ET AL: "RETRIEVAL ALGORITHM EFFECTIVENESS IN A WIDE AREA NETWORK INFORMATION FILTER" PROCEEDINGS OF THE ANNUAL INTERNATIONAL ACM/SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, CHICAGO, OCT. 13 - 16, 1991, no. CONF. 14, 13 October 1991, BOOKSTEIN A;CHIARAMELLA Y; SALTON G; RAGHAVAN V, pages 114-122, XP000239163 see the whole document ---	1-63
A	GOLDBERG D: "USING COLLABORATIVE FILTERING TO WEAVE AN INFORMATION TAPESTRY" COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY, vol. 35, no. 12, 1 December 1992, pages 61-70, XP000334368 see the whole document ---	1-63
A	"INTELLIGENT DOCUMENT ANALYZER FOR SMARTMAIL" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 34, no. 4A, 1 September 1991, page 215 XP000210889 see the whole document -----	1-63

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☑ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.